微软（中国）测试工程部总工程师呕心沥血之作

捉虫记

——大容量Web应用性能测试与LoadRunner实战

施 迎 著



CD-ROM

6.5小时多媒体教学视频

清华大学出版社

程序员典藏大系

捉　虫　记

——大容量 Web 应用性能测试与 LoadRunner 实战

微软（中国）测试工程部总工程师　施迎　著

清华大学出版社
北　京

内 容 简 介

本书主要讲解大容量 Web 性能测试的特点和方法，以及使用业内应用非常广泛的工具——LoadRunner 9 进行性能测试的具体技术与技巧。

本书共 17 章，分为 5 篇。第 1 篇介绍软件测试的定义、方法和过程等内容；第 2 篇介绍 Web 应用、Web 性能测试的分类、基本硬件知识、Web 应用服务器选型、各操作系统性能计数器的获取等内容；第 3 篇介绍如何使用 LoadRunner 进行 Web 应用性能测试，包括 LoadRunner 基础、编写测试计划、配置测试环境、LoadRunner 中的场景、监控图表与函数、执行场景和分析结果等；第 4 篇介绍通用性能测试结果分析及其他性能测试工具；第 5 篇介绍大容量 Web 应用性能测试实战案例及 Web 性能优化等内容。

本书理论结合实践，讲解图文并茂，并且将 IT 技术与生活场景结合起来，生动而又形象。另外，为了让读者更加直观、高效地学习，作者专门录制了大量多媒体教学视频。这些视频收录于本书的配书光盘中。

本书既适合网站测试人员和 Web 应用性能测试人员阅读，也可供其他相关测试人员和大中专院校相关专业的学生学习和参考。

图书在版编目（CIP）数据

捉虫记——大容量 Web 应用性能测试与 LoadRunner 实战 / 施迎著．—北京：清华大学出版社，2010.6
ISBN 978-7-302-22231-6

Ⅰ．①捉…　Ⅱ．①施…　Ⅲ．①主页制作－程序设计②性能试验－软件工具，LoadRunner
Ⅳ．①TP393.092②TP311.56

中国版本图书馆 CIP 数据核字（2010）第 043488 号

责任编辑：夏兆彦
责任校对：徐俊伟
责任印制：
出版发行：清华大学出版社　　**地　　址：**北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185×260　**印　　张：**26　**字　　数：**648 千字
（附光盘 1 张）
版　　次：2010 年 6 月第 1 版　**印　　次：**2010 年 6 月第 1 次印刷
印　　数：1～ 000
定　　价：　　元

产品编号：036229-01

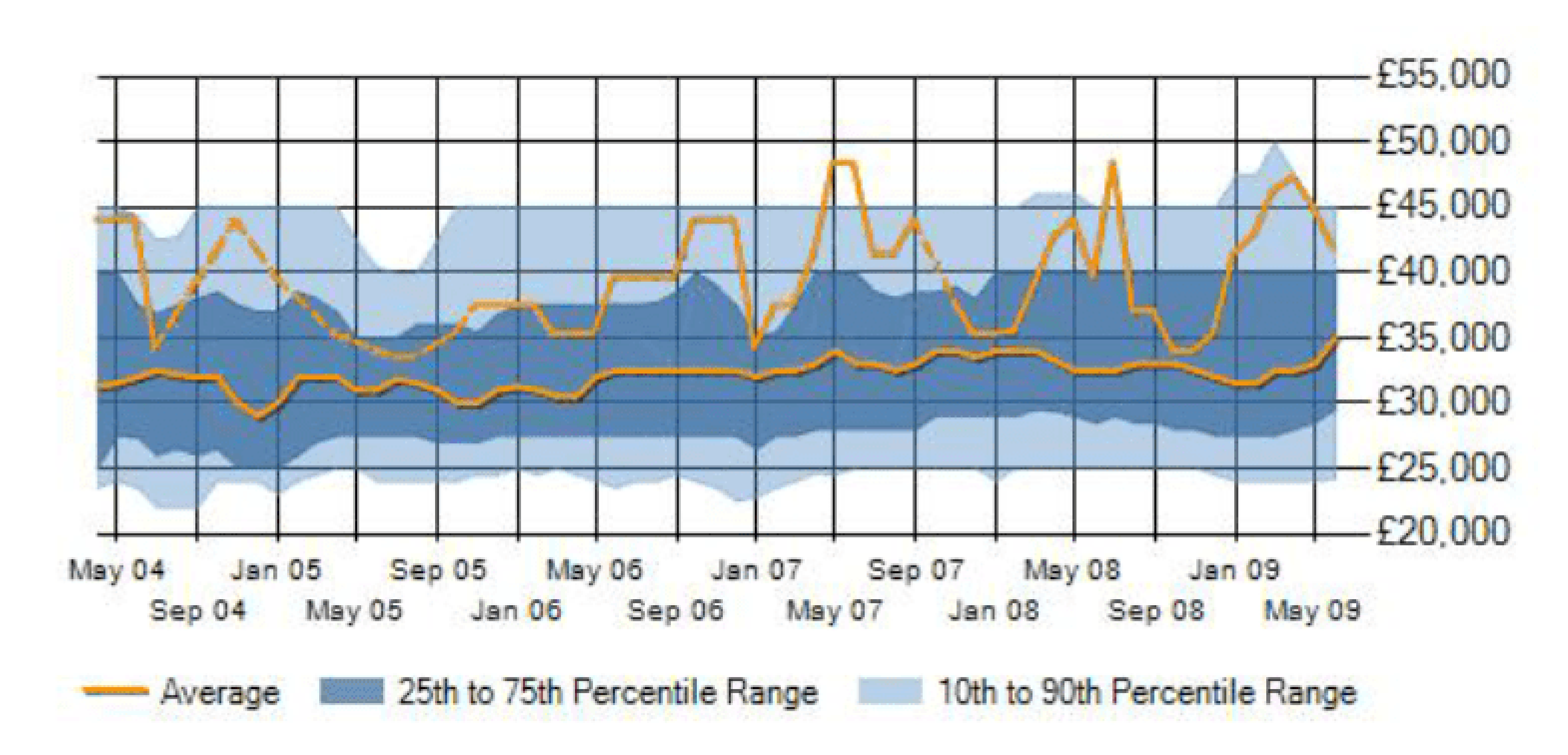
前　　言

不轻言放弃，一切皆有可能！

21 世纪的 Web 应用什么最重要？性能！

进入 21 世纪以来，互联网行业经历了深刻的变革，现在的网站与 10 年前的网站已经今非昔比了。这其中很大的原因是由于人们生活和工作节奏的加快：对于 Web 应用的开发过程，人们希望时间越来越短，新产品新应用能够尽早投向市场，于是采用了快速开发、敏捷开发、项目管理、.NET、ORM（对象关系映射）等诸多技术、规范和平台来保证开发速度与质量。而对于已经上线的 Web 应用，人们又希望它能够更迅速有效地与用户交互。因此，对于 Web 应用的性能要求越来越高。有越来越多的网站和软件开发公司在招聘性能测试人员，而且鉴于工作内容的独特性，起薪较一般测试人员要高。

下图显示了英国计算机职业发展网站，从 2004 年 4 月开始至 2009 年 5 月截止，对测试人员和性能测试人员薪酬进行统计的变化曲线，图上方曲线为性能测试人员薪酬，下方是一般测试人员薪酬。可见，两者的差别还是很明显的。



在国内，近年来软件测试逐渐成为热门的职业，如果再具备性能测试的特色，将对今后的职业发展有很好的促进作用。

性能测试工程师是什么样的人？

作为一名合格的性能测试工程师，他（她）应该具备如下的专业素质：

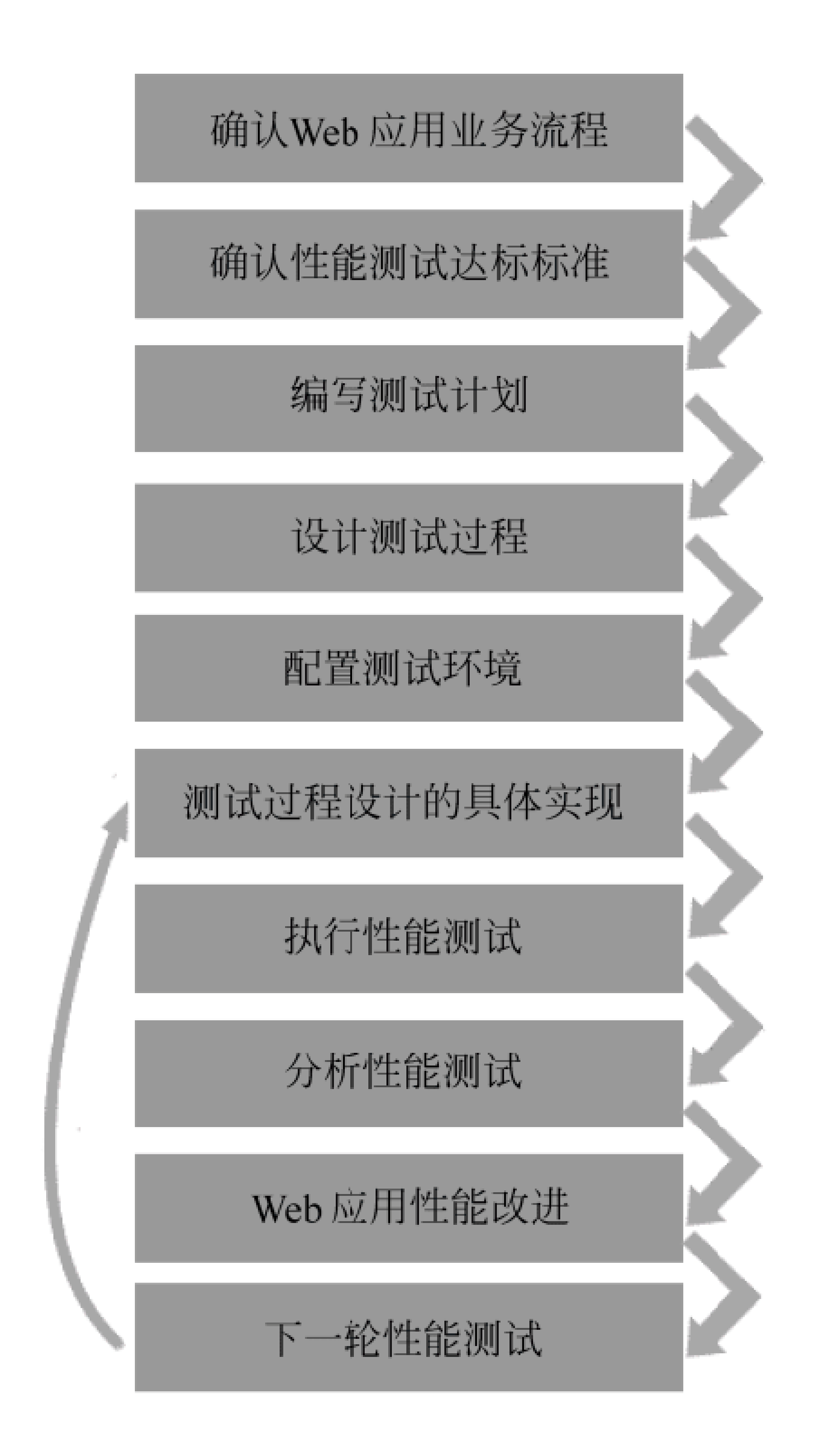
- 软件使用经验丰富，对于软件的不正常行为很敏感。性能测试工程师有时需要要是个“急性子”，反应慢的软件在这里将是“零容忍”。
- 有好奇心，乐于探索软件功能，乐于尝试新的软件产品。
- 乐于探索未知，追根溯源。对于一个 Bug，必须有追根溯源的精神，才能够发现

它的特点，这个性格特征在判断 Bug 的产生原因，以及是否与其他 Bug 重复等日常的工作内容中都会展现。
- 有耐心，不轻言放弃。测试工程师在工作中经常会试图复现一个软件中的 Bug，这需要细心、耐心和坚持。
- 必须具备一定的创造性。测试工程师是无法模拟出用户使用软件的所有场景的，因此必须具备一定的创造性，通过测试更多情况下软件的不同表现，发现被测软件更多的问题。
- 具备一定的沟通和交流技巧。
- 具备基本的数据结构，操作系统等方面的知识，还要有一定的程序开发经验。性能测试工程师要和软件及其所处的操作系统环境打交道，没有前述这些知识是很难在职业发展中获得很大成绩的。

如果你喜欢和这样的人一起工作，或者你就是这样的人，那么很好，经过本书的学习，很快你也能成为他们中的一员。

性能测试的大致过程

描述了性能测试工程师的大致样子，下面再来了解他们所从事的工作内容。性能测试的大致过程如下图所示。



在上图中，有很多步骤与其他类型的测试：比如功能测试等是非常类似的。但是，性能测试的工作内容有自己的特点。

确认性能测试达标标准就是其一：与功能测试清晰的标准不同，性能测试标准需要结

合实际情况和项目阶段、必要时大家讨论制定。本书前几章介绍了如何确认标准，以及业内普遍接受的标准是什么这些问题。

配置测试环境是其二：Web 应用的性能测试环境要尽可能地接近真实生产环境，这才能保证测试数据的有效性。本书介绍了目前很热门的虚拟化技术在准备测试环境中的贡献，具备很强的实用性。

分析性能测试也是性能测试工程师比较独特的工作内容之一。这是由于进行性能测试的重要目的就在于为优化性能服务。实际上，不限于以上列举出的这几点，上图中的每一个步骤本书都有专门的章节介绍，非常全面。

性能测试的乐趣

有不少人有这样的看法："测试工作是枯燥、乏味的，没有什么前途"。笔者在这里却要说这种看法是一种误解。与开发工作相比，测试工作考察的是更为全面的 IT 领域知识，虽然在某一点上不必很深入，但是知识和能力一定要尽可能全面。如果一直用这样的标准要求自己并努力做到，读者会发现经验在逐步地积累，灵感越来越多，最终可以成为资深的专家，完全可以打破"IT 人士吃青春饭"的年龄怪圈。君不见，在微软等大企业的员工之中，有很多年长的高级测试工程师依然在从事自己喜欢的测试工作吗？实践证明，测试完全可以成为一个终身的职业。

从角色来说，测试工程师更是开发人员和最终用户的中间人：在黑盒测试的时候，测试工程师可以说是用户的化身；在白盒测试的时候，测试工程师可以说是开发人员的帮手。同一个身份，能够扮演两种角色，不是很有意思的工作吗？

"快乐地工作，快乐地玩"。性能测试作为各类测试中相对高端的一部分，更是一个可以快乐起来的工作。它能增进读者对软件、操作系统乃至硬件的综合理解，每次运行测试都会学到新的知识，获得新的见解。当亲爱的读者在软件中发现第一个性能方面的 Bug，一定会很激励自己，一定会博得同事和老板的尊重。当亲爱的读者了解到操作系统各组件的原理、优点、不足乃至技术内幕，进而对某些似是而非的流行观点发表独立的看法，甚至指出专家的错误时，一定会有一种专业人士的成就感。当亲爱的读者具备了一定的性能测试经验，形成自己在整个测试职业生涯中的特色，一定能够在竞争激烈的职场中脱颖而出，成为公司内不可替代的人。

总之，性能测试是一个非常具有前途、急需人们去开拓的领域。性能测试，充满乐趣。

如何快速掌握性能测试的技巧——资深测试工程师的心得共享

如何学好 Web 应用的性能测试，除了读专业书籍这一途径之外，不妨听听下面这些资深测试工程师的心得。

试装测试工具软件。借助工具软件的逻辑来侧面了解性能测试，这是学习新技术的不二法门。其实很简单，测试工具软件是为了方便测试的目的开发的，它肯定要遵循测试的规范、术语，采用主流技术。安装测试工具的试用版并实际执行一次简单的性能测试，就好像驾校学车时的第一次上路，是必经的阶段，而一本较好的入门书，则起到了教练的作用。

多看测试工具的帮助文档。这是进入测试领域后深入理解测试工具的捷径。测试工具能做什么？怎么做更有效率？只有仔细浏览帮助文档，才能成为掌握测试工具的高手。

多看“课外书”。所谓课外书，就是指与性能相关的计算机操作系统、数据结构、计算机语言等方面的书籍。性能是 Web 应用综合实力的体现，硬件、软件、网络都会对性能有影响，因此，在成为高手的路途中，光拘泥于测试工具是不够的。

成为测试高手的标志是：

脱离了测试工具的束缚，结合实际工作，开发适用于本地的测试工具。测试工具有一定的局限性，而且，每一个 Web 应用也有自己的特点，期望一个工具作为万灵药是不可能的。这一个步骤是最难的，因为它需要对测试与开发都有很好的经验与理解。在开发自定义测试工具的道路上，会遇到很多的困难，而一旦克服了这些困难，就可以称为测试的真正高手。好比武林中的“飞花摘叶，俱可伤人”。

如果能完善自定义的测试工具，使之成为平台，就能够适应更多的情况。能做到这一步，又可以称为高手中的高手，正是“无招即有招”。

本书有何特色？

性能测试的知识、方法较繁杂，与软硬件的结合也较紧密，作为初学者，能否在有限的时间内快速学好呢？这恐怕是每一位朋友在读本书之前都要问的问题。

为了方便读者阅读和学习，本书精心地安排了各章内容，由浅入深，循序渐进，具备如下鲜明的特点：

- 本书覆盖面宽，涉及性能测试的诸多方面，如理论、软硬件选型、测试报告编写技巧、测试数据分析、虚拟化技术在测试中的应用等，相比同类书籍中较为全面。
- 书中充分考虑到一般测试与性能测试的区别，着重培养读者将软硬件相结合来考虑性能问题的习惯和意识。
- 书中所有实例都采用图示、文字详细说明相结合的方式，做到了明确直观。
- 对于菜单和软件功能的介绍，并不限于按照菜单顺序，而是具备一定的逻辑性，循序渐进，有利于读者理解。
- 对每个需要注意的识点，如核心要点、使用技巧等，都特别注明，使阅读者能够明确重点。
- 内容不限于介绍 LoadRunner 本身，有利于读者脱离某个工具软件的束缚，真正将性能测试的知识掌握得更好。
- 书中介绍的软件版本较新，如 LoadRunner、VirtualBox、VisualStudio 等软件均为截至本书完稿时的最新版本，保持了时效性。
- 本书理论结合实践，讲解图文并茂，步骤详细而直观，并且将 IT 技术与生活场景结合起来，阅读起来生动形象。
- 另外，为了让读者更加直观、高效地学习，作者专门为本书录制了大量多媒体教学视频。这些视频收录于本书的配书光盘中。

希望读过本书之后，读者都能够自豪地对开头的问题做肯定的回答：“性能测试知识，我真的可以有！”

本书的知识结构体系

本书共 17 章，分为 5 篇。

第 1 篇　Web 测试背景知识（第 1～3 章）：本篇主要介绍了软件测试的定义、方法和过程等基础知识，为读者打下一个测试领域的良好基础，为性能测试做准备。

第 2 篇　Web 性能测试入门（第 4～6 章）：本篇主要介绍了 Web 应用和 Web 性能测试的诸多分类，除此之外，还详细讲解了实用性较强的基本硬件知识、Web 应用所用服务器的选型方法、各操作系统（Windows、Mac OS X、Linux 等）性能计数器的获取等重要内容，以便让性能测试领域的新手也能快速入门。

第 3 篇　使用 LoadRunner 进行 Web 应用性能测试（第 7～13 章）：本篇基于之前的性能测试知识，全面讲解了 LoadRunner 9 的使用方法。本篇可分为 3 小部分：通过脚本模拟单个真实用户行为；通过场景模拟多个真实用户的行为及通过分析器进行性能分析。同时，按照性能测试的流程，本篇对部署和准备测试环境、撰写测试报告等都进行了专门讲解。

第 4 篇　Web 性能测试提高（第 14～15 章）：本篇主要介绍了通用性能测试结果分析方法和其他性能测试工具，便于测试工程师可以不受 LoadRunner 工具的限制，拥有更广阔的视野。

第 5 篇　Web 性能测试实战（第 16～17 章）：本篇主要介绍了大容量 Web 应用性能测试实战案例及 Web 性能优化等内容。本篇是对本书内容的全面总结，便于进一步深化读者所掌握的性能测试的观念、理论和方法。

本书适合哪些读者阅读？

- 网站测试人员；
- Web 应用性能测试人员；
- 广大性能测试爱好者；
- 想学习 LoadRunner 的人员；
- 想进入软件测试行业，并希望拥有特色知识结构的职场新人；
- 已经工作，希望有所提高的初级性能测试工程师；
- 目前从事一般测试工作，但希望转做性能测试的人员。

本书作者及编委会成员

本书主要由微软（中国）测试工程部总工程师施迎编写，其他参与编写和资料整理的人员有陈世琼、陈欣、陈智敏、董加强、范礼、郭秋滟、郝红英、蒋春蕾、黎华、刘建准、刘霄、刘亚军、刘仲义、柳刚、罗永峰、马奎林、马味、欧阳昉、蒲军、齐凤莲、王海涛、魏来科、伍生全、谢平、徐学英、杨艳、余月、岳富军、张健和张娜。在此一并表示感谢。

本书编委会成员有欧振旭、陈杰、陈冠军、项宇峰、张帆、陈刚、程彩红、毛红娟、聂庆亮、王志娟、武文娟、颜盟盟、姚志娟、尹继平、张昆、张薛。

目　　录

第 1 篇　Web 测试背景知识

第 2 篇 Web 性能测试入门

第 4 篇 Web 性能测试提高篇

第 5 篇 Web 性能测试实成

第 1 篇　Web 测试背景知识

第 1 章　什么是软件测试

在我国宋代，有一位叫宋慈的法医学家写了一本《洗冤集录》。在书中，他讲述了很多断案的经验，其中有一个用银针验毒的方法至今仍广为流传。比如在很多电视剧中，我们能经常看到皇帝在进膳的时候，由于害怕被人暗害，总要让可怜的太监或者宫女先用银筷子尝上几口饭菜，没有出现问题再正式用餐。这种用银针进行的试验就可以说是一种测试的雏形吧，银针充当了测试工具，而太监或者宫女就是古代的测试工程师。

时光飞逝。随着科技的发展，我们生活周围有了越来越多的产品，它们在出厂销售前都要进行测试，不仅要保证功能完好，还要确保对使用者的伤害在允许范围内。因此在工厂里，逐渐出现了这样一个部门，由它来负责检验产品，被称之为质量检验或者质量保证部。

上个世纪中后期，软件出现了，它作为人们日常生活中天天都会使用的产品，同样也需要质量的保证。有一种误解：软件的质量问题并不那么重要，比如 Windows 等操作系统，各种桌面的应用软件，像 IE 浏览器，如果它出现了问题，程序会失去响应甚至严重的系统会蓝屏，那么只需要在任务管理器中将它删掉就可以了，最多重新启动电脑，一般都能够继续使用。这只是一方面，另一方面，有很多非常重要的软件在我们看不见的地方默默地运行着，如果它们出现了问题，影响就很大了。

为了说明软件质量的重要性，这里举一个比较著名的软件质量造成的事故。

1962 年，美国的航海家 1 号（Mariner 1）火箭升空，由于控制火箭的软件出现问题，直接导致火箭升空后因偏离轨道而被迫引爆，造成当时 1800 万美元的损失。事后查明，是程序员在编写软件代码时，误写了其中一个公式的上标造成轨道计算失误的。

因此，软件公司也需要质量保证部门。我们把该部门的组成人员称为 QA 工程师，QA 即 Quality Assurance 质量保证的简称。软件是否符合质量是通过测试来验证的，因此他们也被称为软件测试工程师。在本书中您即将遇到的各种行为，绝大多数都将是软件测试工程师在工作中所要实现和完成的。

1.1　软件开发的基本知识

对于每一位进入软件测试行业的新人来讲，公司的入职培训是一个很好的学习机会。

1.1.1　软件开发公司技术部门的基本结构

可将软件测试部门类比于工厂车间的质量保证部门，那么显而易见，如果在工厂中要做好质量控制的工作，必须熟悉本厂生产的产品和流程。换句话来说，作为软件生产的参与者，了解被测试的软件也是非常重要的一件事情。这也正是经理要求小白在短期内尽快

熟悉的内容。

【什么是软件】

中国大百科全书中对软件的定义是：软件是计算机系统中的程序和相关文件或文档的总称。软件是从英文 Software 翻译过来的名词，与硬件（Hardware）相对应。

因此，软件开发公司就是制造这些程序和相关文件或文档的商业机构。一般来说，软件开发公司的技术部门由几个子部门或者角色组成：开发部门、测试部门、部门经理或者项目经理，另外有的公司还有技术支持部门。对应于传统行业，分别相当于生产车间，质量控制部门，部门经理和售后服务部门。如表 1-1 列出了常见软件开发公司技术部门的不同职责。

表 1-1　常见软件开发公司技术部门的角色分类

部　　门	职　　责
软件开发部门	开发软件：确定软件实现方法，编写软件程序代码
软件测试部门	测试软件：确定测试方法，编写自动测试软件的代码，手工测试软件，记录并跟踪软件 Bug
技术总监或项目经理	在所属或其他部门之间沟通，协调项目或者开发测试进度，为成员提供各种资源
技术支持部门	软件开发完成后在客户处部署产品，并解决与反馈使用中出现的问题

Web 应用是一种特殊的软件。那么开发 Web 应用的网站与一般的软件开发公司有什么不同呢？

对于小白所处的商业网站来说，网站程序和相关文件或文档也可以称之为软件，其技术部门的结构也和软件开发公司基本类似，但是各部门日常工作的方式则有所不同。

- 商业网站每天都要有很多页面的更新，每次更新后当时浏览网站的人立即可以看到；而软件开发公司一般一年或者几年推出一个产品，在产品没有上市的时间内，用户只能使用旧的版本。也就是说网站软件的变化要比软件开发公司频繁，网站软件的开发与用户使用处于同一时间段内。
- 商业网站以服务器为核心，网站软件主要运行在服务器上；而软件开发公司的产品主要运行在用户的电脑上。

【演唱会与专辑】

商业网站与软件开发公司的运作模式有点类似于歌手开演唱会和发专辑的区别。在演唱会上，歌手与观众的互动性更强，每一个细节的变化也都能被观众捕捉到；而歌手专辑则相当于软件产品的某个版本，是提前制作完成之后再上市销售的。

1.1.2　软件危机

小白在熟悉了技术部门的大致结构，网站与软件开发公司的区别之后，经理开始介绍和软件测试相关的背景知识。软件危机就是产生软件测试这一职业的重要推动力之一。

从 20 世纪 50 年代以来，软件的规模越来越大，复杂性也越来越高，另外，在 20 世纪 80 年代，伴随着计算机的普及和应用需求的飞速增长，互联网开始蓬勃兴起。现在，现代人的生活已经越来越依赖各种各样的软件，软件不再是大学实验室里科学家的工具，而

成为我们生活的一部分。从操作系统，比如每一台个人电脑所安装的 Windows XP 或者 Vista 系统，到小小的桌面程序——一个简单的连连看小游戏，再到 Google 网站上可以编辑的在线文档工具，软件的开发、管理、维护的复杂性和高成本现象也日益突出，在某一段时期，暴露了很多问题。因此在 20 世纪，有人提出了“软件危机”的说法，来说明这种现象。

【软件复杂性的类比】

其实我们中国人是很容易理解软件复杂性的。一些在人口少的国家不成为问题的问题，放在十几亿人口的环境中，就会产生不大不小的麻烦，比如每年的春运——需要火车票的人是如此之多，而火车的座位是固定数量的；需要回家的人的分布是如此之广，而火车站的位置也是固定的。依此类比，当软件的代码量越来越庞大，要满足的需求越来越广泛时，出现局部的危机是很容易理解的。

1.1.3　软件危机的几个体现

随着软件越来越复杂，质量越来越难于控制，于是出现了所谓“软件危机”。具体而言，软件危机有以下几个体现。

- 软件需求的增长无法快速得到满足。这一点在前文已经有所讲述。
- 软件生产成本变高，价格越来越昂贵。软件的代码量增加，所投入的人工成本，也就是软件开发相关人员的成本也会增加，还要增加采用各种新技术的成本等。
- 软件生产进度难于控制。
- 软件的用户需求不容易定义。这一点也很重要：目前绝大多数的软件已经不止满足单一的需求，因此用户真正所需要的不一定能够完美地实现。
- 软件质量不容易保证。这一点也是由软件复杂度的增加而增加。还是举春运的例子，如果火车上人人都有座位，那么每个人的心情都会很好；但如果到处挤满了人，每位旅客回家的心情总会受到影响，从而影响对列车服务的评价。软件质量和用户的评价同样是相关的：经常造成死机、异常退出或者按钮单击后没有反应的软件，很难说是质量好的软件。
- 软件可维护性变差。软件同样是需要维护的：一方面是对于用户使用过程中的维护，这一功能由客户服务或者技术支持部门来完成；另一方面是对于软件本身代码和文件文档的维护，这一功能由开发部门或者测试部门来完成。随着软件的日益庞大，软件本身经历的修改越来越多，管理维护软件的各种版本变得日益困难。

由于软件危机有这么多的影响和危害，所以促使人们静下心来研究软件开发过程中的规律，这就产生了软件生命周期的概念。

1.1.4　软件生命周期

软件生命周期，英文为 Software Lifecycle，就是软件开发、使用和消亡的过程，具体而言，包含软件需求分析、软件设计、软件实现与测试和软件发布、部署与维护这 4 个过程。

在商业软件开发公司内部，人们往往遵循一定的软件生命周期模型，这样和被开发软件相关的所有人员都按照这个模型的标准或者步骤开展工作，统一行动，有助于提高生产

效率，从而减少沟通和实施的成本，获得更大的商业利益。而对于软件生命周期的不同理解和划分，就形成了不同的软件生命周期模型。

1.1.5　常见的软件生命周期模型

目前来讲，主要的软件生命周期模型有如下几种。

- Big-Bang：大爆炸模型。
- Waterfall：瀑布模型。
- Spiral：螺旋模型。
- Code and Fix：边做边改模型。

由于本书并不是以软件工程为探讨内容，因此在这里只通过人们过河的类比来简单介绍一下前述这几种软件生命周期模型的特点。

小学课本里有个寓言叫做“小马过河”，小马在过河前遇到了不同的小动物，它们对于河水深度的理解是不同的，会导致小马过河时的不同选择，参见图 1-1。假设把待开发的软件产品比喻为小马面前横着的那条小河，那么开发软件的过程也就是过河的过程，那么如何过河就会有不同的结果。



图 1-1　小马过河：对河深度的理解影响过河的方法

1.1.6　直接冲过河去的大爆炸模型

大爆炸这个名称来自于天体物理有关宇宙形成方式的一种理论：宇宙是在亿万年前的大爆炸中诞生的。与此类似，软件开发公司把金钱、办公场地和人员全部投入到一个产品的开发当中，经过一段时间，产品出炉，这样的形式就是大爆炸模型。

大爆炸模型的优点就是简单，没有很多的软件设计，对项目的管理也很少，目前不少小公司由于各方面的限制不得已或者不自觉地采用了这样的开发模型。但是它的优点也造成了它的缺点：开发出来的软件质量不可控制。

在这样的模型中，由于没有周密的计划，软件测试往往是在产品即将上市的前夕才开始，在很多公司中甚至没有专职的测试工程师，由开发人员或者其他人员代劳，因此测试人员面对的产品与客户、使用者要面对的产品基本一致。从前文所述可以得知，在这样的阶段发现 Bug，返工修改代码的代价是非常大的。

回到过河的比喻中来，大爆炸模型就相当于小马先退后几步，集中精力和能量，然后快速冲过去。这样的结果取决于河的宽度和深度。如果软件非常复杂，很可能过河的小马半途就淹死了，无法到达对岸。

1.1.7　摸着石头过河的边做边改模型

边做边改模型比起大爆炸模型来说进了一步。在开发软件产品的开始阶段，先有一个大概的设计，然后开始编码，测试，发现 Bug，修改 Bug 这样的循环，直到整个产品的轮廓日渐清晰，最终完成产品。用一句俗话来描述，就是“摸着石头过河”的过程：先以河里的一些石头为支点，走入河道，再经过不断的试探和返回得到一条路线，最终到达目的地。

由此可见，边做边改模型中测试的参与要比大爆炸模型中要早得多，而且也重要得多。边做边改模型的优点就是适用于某些中小型项目的快速开发，软件产品的成果也会在最早的阶段显现出来：和在岸边冥思苦想如何过河的人相比，先站在河道里的石头上，总是让人看到更多的希望。

【边做边改模型被较多采用】

这种开发模型被大多数公司所采用，是大多数测试工程师在实际工作中最常遇到的开发模型之一。而且，它和最近几年很流行的敏捷开发也有一定的关系。

1.1.8　制定周密过河计划的瀑布模型

从现在开始，下面的这两个模型就不适合小马了，只有人和外星人才有这样的能力。如图 1-2 介绍了软件开发的瀑布模型，由于图中的箭头好像瀑布的水流，从上至下，因此得名。

回到过河的例子中来，瀑布模型过河具备如下特点：

- ❑ 过河前，首先花费大部分的时间对河进行详细的勘察，选择合适的下水点，选择合适的过河工具，制定详细的分步骤过河计划。
- ❑ 一旦过河计划制定，将不会大更改，开始过河。在河中完全按照计划进行，无法返回起点。这也是为什么称此模型为瀑布的原因，瀑布是飞流直下三千尺，想从下面返回瀑布的顶端，何其难。

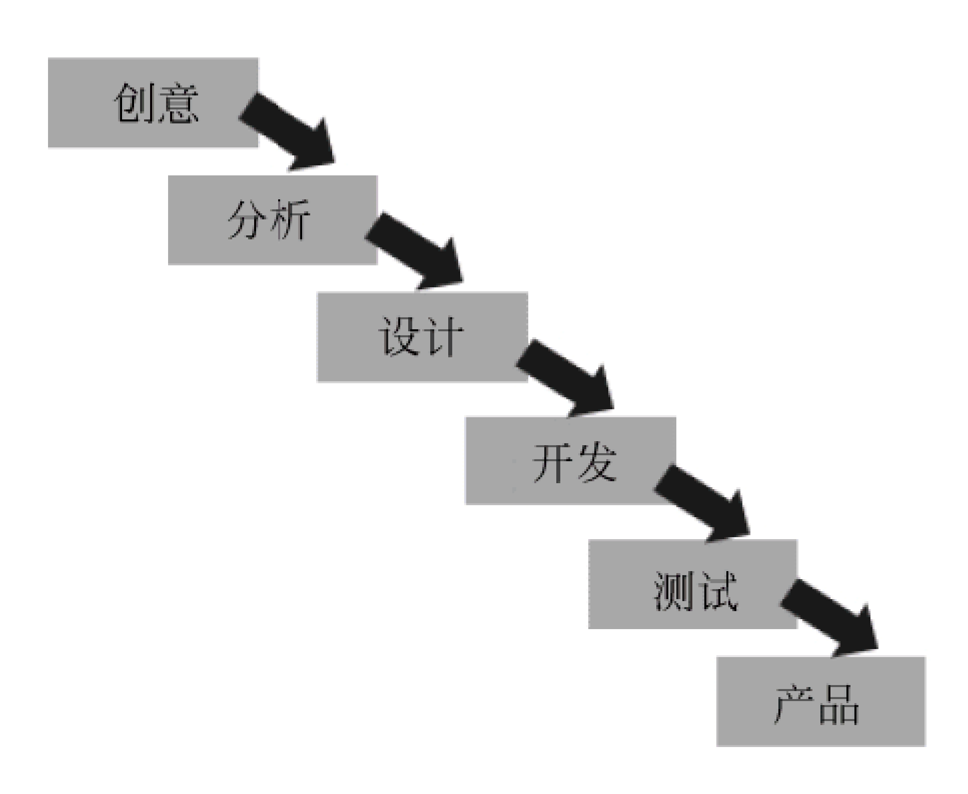


图 1-2　瀑布模型示意图

- ❑ 在每步骤即将完成时，都会对这一步骤进行总结，如果进行下一步骤的条件不具备，将停留在原地，等待条件具备。

瀑布模型看起来给人很专业的感觉，所以，对于软件开发人员有比较高的要求。

- ❑ 要对待开发的软件（或者要过的河）有细致、全面、准确的了解。如果理解错误，将导致计划失败，没有返回重来的机会。
- ❑ 职业素质、职业纪律要比较高。软件开发人员要具备坚定执行计划的能力。

这种要求也就产生了瀑布模型的缺点，那就是无法完美适应当今要求快速开发产品，从而占领市场的软件行业现状。因为制定详细的、理解完整的计划很难，聚合很多专业的开发人员有时候也很难，而市场对于软件更新换代的要求期限越来越短。为了适应变化，人们又提出了螺旋模型。

1.1.9　计划赶得上变化的螺旋模型

前文提到，为了适应计划和变化两方面的因素，螺旋模型被提出。螺旋模型的示意如图 1-3 所示。可以看到，它的确很类似一个螺旋。

与边做边改模型类似，螺旋模型也具有循序渐进的特点，对软件最终实现什么不一定有完全确定的理解，而是摸着石头先下水。但是在选择过河的每一个石头前经过了周密的计划和考虑，从这一点看，又类似瀑布模型。可见，螺旋模型实际上是边做边改模型和瀑布模型的有机结合。螺旋模型有如下 4 个步骤。

（1）确定项目目标、可用资源、各种实现的方法，项目的各个阶段。

（2）在某个阶段中，确认、解决当前阶段项目进展中出现的风险。

（3）评估各种方法，开发、测试代码，实现当前阶段的目标。

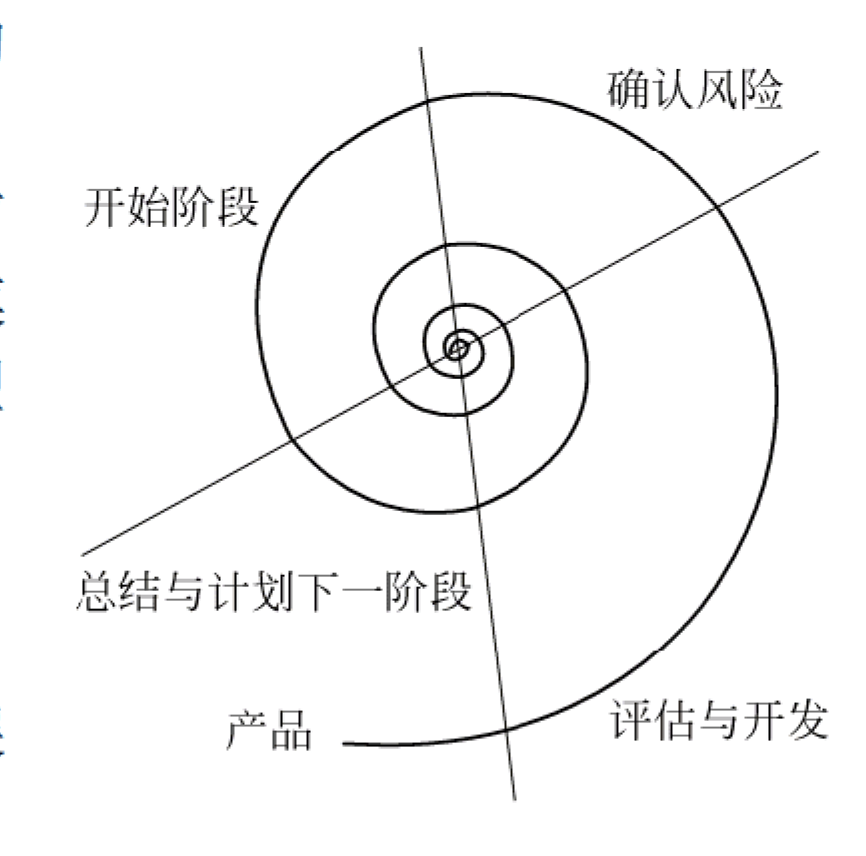


图 1-3　螺旋模型示意图

（4）总结当前阶段，计划下阶段的目标和实现方法，重复第（2）步。

在图 1-3 中螺旋线被两条直线划分成 4 个部分，分别是上述的 4 个步骤。在每一步骤中由于被直线切割会有多段曲线，每一段曲线就代表了在不同阶段中所进行的相同某个步骤。

【螺旋模型的优点】

由此可见，螺旋模型是多次计划，边做边改，这样既保证了软件开发任务的清晰，也降低了开始一次计划，因为理解不完整或者市场变化后导致项目失败的可能性。

1.1.10　4 种模型的总结

前文讲述了 4 种软件开发模型，那么在具体项目开发中采取哪一种最好呢？答案是它们各有利弊，需要灵活采用。这几种开发流程的优缺点比较如表 1-2 所示。

表 1-2　4 种软件开发流程的优缺点

开发流程分类	优　　点	缺　　点
大爆炸模型	简单，不用学习就会	拍脑门的想法，产品质量无法保证。尽量避免使用
边做边改模型	快速得到可运行的版本	计划有些缺乏，导致版本前后变化较大。可选择的模型之一
瀑布模型	计划周密，专业，按部就班实现	相对难于做到快速开发，以抢占市场。可选择的模型之一
螺旋模型	计划变化同时考虑	可选择的模型之一

当然，在几十年的软件开发过程中，人们还提出了很多其他的开发模型，不过，作为测试工程师，我们对这几种主流模型有所了解就可以了。进一步深入的内容并不是本书所讲述的范畴，读者可以参看软件工程的相关书籍。

1.1.11　软件开发的几个阶段

不管采用哪一种开发模型，按照时间顺序，所有的软件开发项目都要经历如下 4 个阶段。

（1）项目启动阶段：了解客户需求、配置相关资源。

（2）项目设计阶段：明确客户需求，确立软件开发、测试的方法。

（3）项目执行阶段：开发与测试阶段。

（4）项目竣工阶段：软件的上市、后期维护与技术支持。

这一分类很好理解，下面再结合小白的工作场景，进行展开介绍。

（1）项目启动阶段。这一阶段一般技术人员参与较少，主要是市场部门，销售部门，技术总监、项目经理等角色的参与：项目成本是多大，开发人员有多少，测试人员有多少，完成时间在什么时候等。

（2）项目设计阶段。这一阶段主要参与者就是需求分析人员、开发人员、项目经理和小白这样的测试人员了。主要目的是确定软件该如何做，做什么：开发人员利用何种技术开发，测试工程师该如何测试该软件，客户如何使用该软件等。这些问题都要确定，形成各自的开发文档、测试文档和需求文档等。

（3）项目执行阶段。开发、测试以及对其的管理就是执行，这一阶段的参与者是开发人员、测试人员和项目经理。开发人员编写程序代码，进行单元测试；测试人员编写测试代码、测试用例，进行功能测试等多种测试。项目经理控制进度，协调各种资源，与设计人员沟通等。

（4）项目竣工阶段。当项目执行完毕的时候，依然要进行部署、软件光盘生产、客户支持、升级补丁包开发和测试等多项工作。这阶段主要的参与者是项目经理，少量的开发人员和测试人员，售后技术支持人员、客户服务人员等。

1.1.12　软件发布的方式

按照目前的软件发布方式，一般有 RTM（Ready To Market，市场发布）、RTW（Ready

To Web，在网络上发布）、RTO（Ready To Operation，可以运营）等多种方式。

- RTM 方式需要在工厂进行光盘的复制生产，用户购买光盘后安装。大部分的操作系统和应用软件采用这种方式的比较多。
- RTW 方式需要在网络上提供下载链接，一般的软件升级包或者游戏软件采用这种方式的比较多。
- RTO 方式则很简单，在服务器上部署软件产品，用户购买其中的某项或者多项服务即可，这是大部分网站或者在线游戏采用的方式。

1.1.13　项目管理与甘特图

前面提到软件项目的流程很重要，那么这种流程的控制一般是由项目经理来完成的，他或者她所从事的这个工作叫做项目管理。

小白所在的技术部门一般一周都要开一个例会，在会上，开发部门、测试部门和项目经理都要对上一周各自所做的工作进行一番总结，安排下周将要做的工作。在这样的例会上，同事们经常会查看项目的进度图来进行讲解，他们把这样的进度图称为甘特图（Gantt Chart）。

如图 1-4 显示的是软件开发过程中常用的项目管理工具 Microsoft Office Project 软件（在这里是 2003 版本，最新为 2007 版本）运行的界面，在其中我们可以清楚地看到软件生命周期中的各个子任务的时间分配，负责人员和项目进度的甘特图。

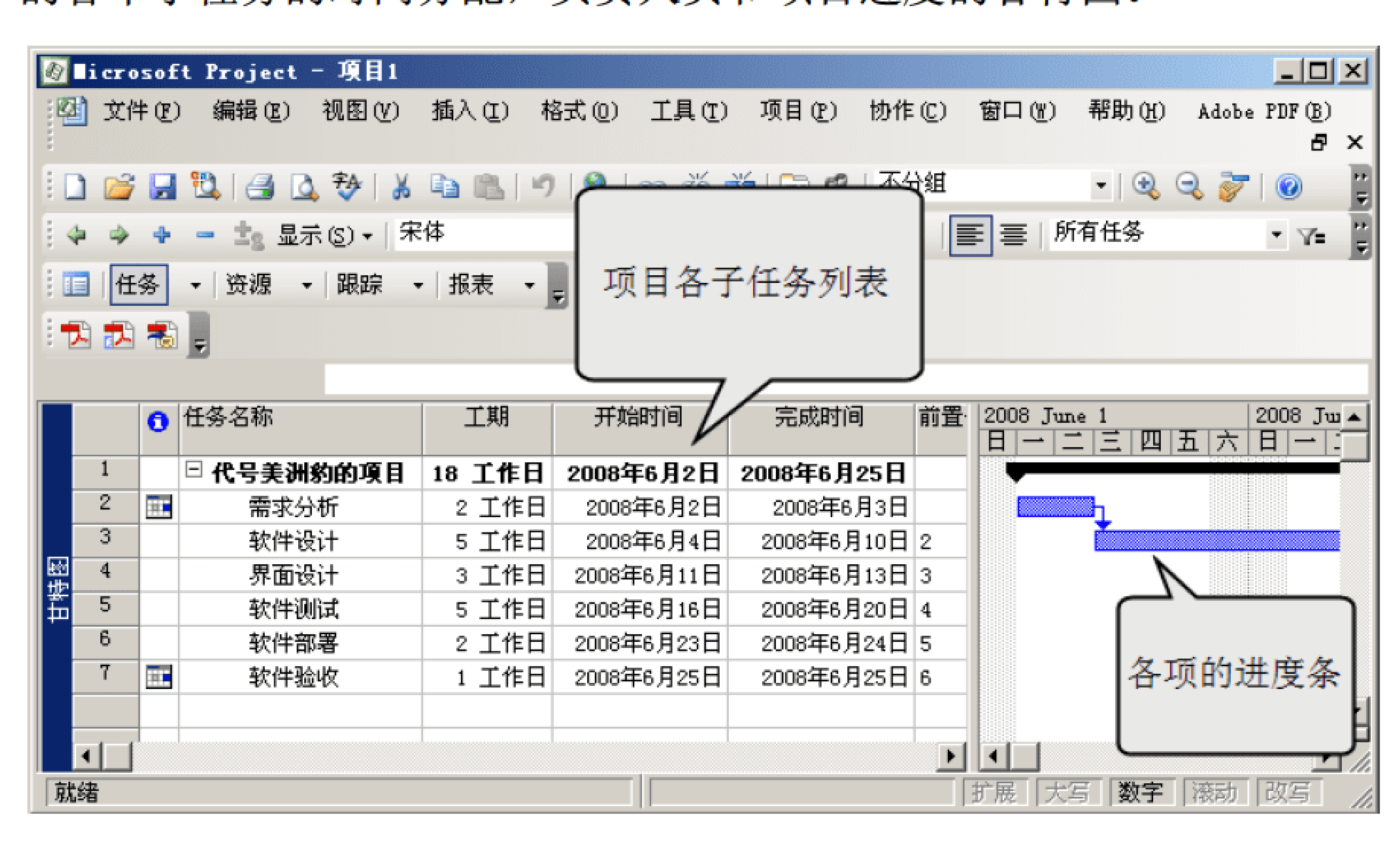


图 1-4　Project 2003 中的甘特图

【甘特图的来历】

甘特图的名称由发明者亨利·劳伦斯·甘特（Henry Laurence Gantt，1861—1919）而来。甘特早年从事的是电气工程师的职业，后来转而从事管理业界的咨询。甘特图是他在晚年发明的一种用于显示项目计划和进度的图表。在诞生的初期，甘特图就被誉为 20 世纪 20 年代的最重要发明之一，广泛应用于一系列的大工程之中。比如 1931 年前后修建的美国胡佛水坝（如果你看过 2007 年的热门电影《变形金刚》，那么对关押威震天的那个水坝

应该有印象，它就是胡佛水坝）。在软件开发领域，很多公司也应用甘特图这一工具来进行项目管理，比如著名的微软公司。

1.2　关于虫子的故事

在熟悉了公司的结构、开发流程，参与了部门例会之后，小白要开始从事具体的软件测试工作了，对于他来说，这一领域陌生而令人兴奋。

在刚上班的一周内，小白不断地听到周围的测试工程师高兴得喊道："又发现 Bug 了！"，看着他们那兴奋的样子，小白也有点跃跃欲试，想赶紧在捉虫的战场上大展身手。那么，什么是 Bug 呢，它为什么这么重要，发现 Bug 为什么这样兴奋？

1.2.1　虫子的来世今生

在本章的序幕部分，我们已经了解了很多由于软件代码的问题使得事情失败的案例了。它们有的后果真的很严重，甚至能够造成对生命的威胁。这肯定不是软件设计者和开发者想要达到的目标，因此，出现这样的情况可以说是软件的错误。

细细的分起来，软件的错误有如下几个词语来描述：

缺陷、偏差、错误、问题、事故、异常。在这一堆词语当中，除了偏差之外，其他的词语所造成的后果给人的感觉都相当严重。所谓偏差，就是软件在使用过程中，和软件设计说明（product specification）所不一致的行为。

那么为什么将这样的软件问题称为 Bug 呢？这里面还有一个故事。

【史上第一个软件 Bug】

该词的原意是"臭虫"或"虫子"。1947 年 9 月 9 日，正值计算机刚刚被发明的时候，哈佛大学的某个计算机实验室正在做实验。由于当时的原始计算机由很多庞大且昂贵的真空管组成，运行时会产生光和热，在下午 15 点 45 分的时候，一个飞蛾（英文是 Moth）钻入了真空管内，导致整个计算机无法工作。当把这只小虫子从真空管中取出后，计算机又恢复正常。后来，虫子的泛称 Bug 这个名词就沿用下来，而那个被拍死的飞蛾也成为了历史上发现的第一个 Bug。

【Bug 渗透到日常生活中】

一般来说，拥有一定知识产权的产品的错误都能称之为 Bug。这方面有一个我们比较熟悉的例子就是电影。影迷们经常议论某热门电影中出现了所谓的"穿帮"镜头，比如在描述古代武侠的影片中天空掠过一架飞机，主角刚才是右脸有伤痕，过一会变成左脸等。这样的镜头也可以说是 Bug，甚至还有专门的网站来记录这些影迷的细心发现，比如 http://www.chinabug.net。

1.2.2　软件 Bug 的 5 个要素

前文笼统解释了软件 Bug 是软件的错误或者偏差。那么在具体的工作中，小白如何判断软件的行为是 Bug 呢？说来简单，根据软件设计阶段形成的功能说明书，英文为

Specification Document，一般简称 Spec。对于具体的判断标准，经理介绍了如下 5 个要素：

- 软件没有实现说明书中所列出的功能。
- 软件出现了说明书中提到不应出现的事情。
- 软件实现了说明书中没有提到的功能。
- 软件没有实现说明书中没有提到但应该实现的功能。
- 软件非常难于学习、使用，运转速度很慢，用户认为无法达到预期。

为了充分理解上述 5 个要素，小白自己打开了 Windows 系统中最简单的一款软件 Notepad，也就是我们平时“不屑于”用到的记事本程序，开始了自己的思考。

1. 软件没有实现说明书中所列出的功能

对于“软件没有实现说明书中所列出的功能是 Bug”这一点是比较好理解的。如果打开记事本软件，却无法在其中输入汉字，或者输入了文本，无法保存成文件，那么肯定是一个很重要的 Bug。

2. 软件出现了说明书中提到不应出现的事情

对于第 2 点，“软件出现了说明书中提到不应出现的事情也是 Bug”，这一点和小白的性能测试工作有相对更紧密的关系。小白要测试的是公司的网站，它要求用户在浏览网站时显示页面尽可能地快，如果超出 5 秒钟则认为是不可接受的。这个“超出 5 秒钟”就是说明书中提到不应该出现的事情，实际出现后肯定是一个 Bug，需要开发人员找出哪里耗费了页面显示时间。

在记事本程序中，如果程序保存文件时出现了程序崩溃（Crash）现象，即属于此类。

3. 软件实现了说明书中没有提到的功能

软件实现了说明书中没有提到的功能也是 Bug 这一点可能有点难于理解。一个软件，功能难道不是越多越强大吗？其实不尽然，实现额外的功能有如下几个缺点，如表 1-3 所示。

表 1-3　软件实现说明书中未提到功能所带来的问题

缺　　点	说　　明
代码量增大	由于代码可能相互影响，因此这部分额外的功能可能对其他功能的实现造成影响，带入新的 Bug
增加额外的开发、测试时间	在软件项目时间固定的情况下，导致投入到其他必备功能的开发测试时间减少，可能影响它们的完成质量
增加了成本，与软件的宣传不完全符合	虽然用户对于增加功能一般不会有意见，但可能影响了公司的销售策略和市场定位

4. 软件没有实现说明书中没有提到但应该实现的功能

小白一般是将网上找到的有用文档保存在随身携带的 U 盘中。这一次，他在测试记事本程序的时候，同样打算将文件保存在 U 盘上，可是由于连日来的文档太多了，优盘已经没有空间，记事本提示无法保存，同时系统托盘有提示说磁盘空间已满。在这种情况下记

事本的行为，就属于实现了说明书中没有提到却应该实现的功能（在磁盘满的情况下，给用户以提示）。如果没有提示，不符合绝大部分用户的使用习惯，也是一个 Bug。

5．软件难于使用、性能差

软件是拿来用的，再好的界面使用不方便也不会产生多大效果。一个网站如果半天都打不开，很难想象还会有多少用户会访问它。因此这样的问题也是 Bug，而且对于性能测试来说，这一个规则很重要。

1.2.3　发现虫子的危害

既然软件 Bug 对产品造成了这么多的影响，那么发现它就显得非常重要了。业内人士都认为，在软件生命周期内的不同阶段发现 Bug，所节省的成本是不同的，如图 1-5 所示。

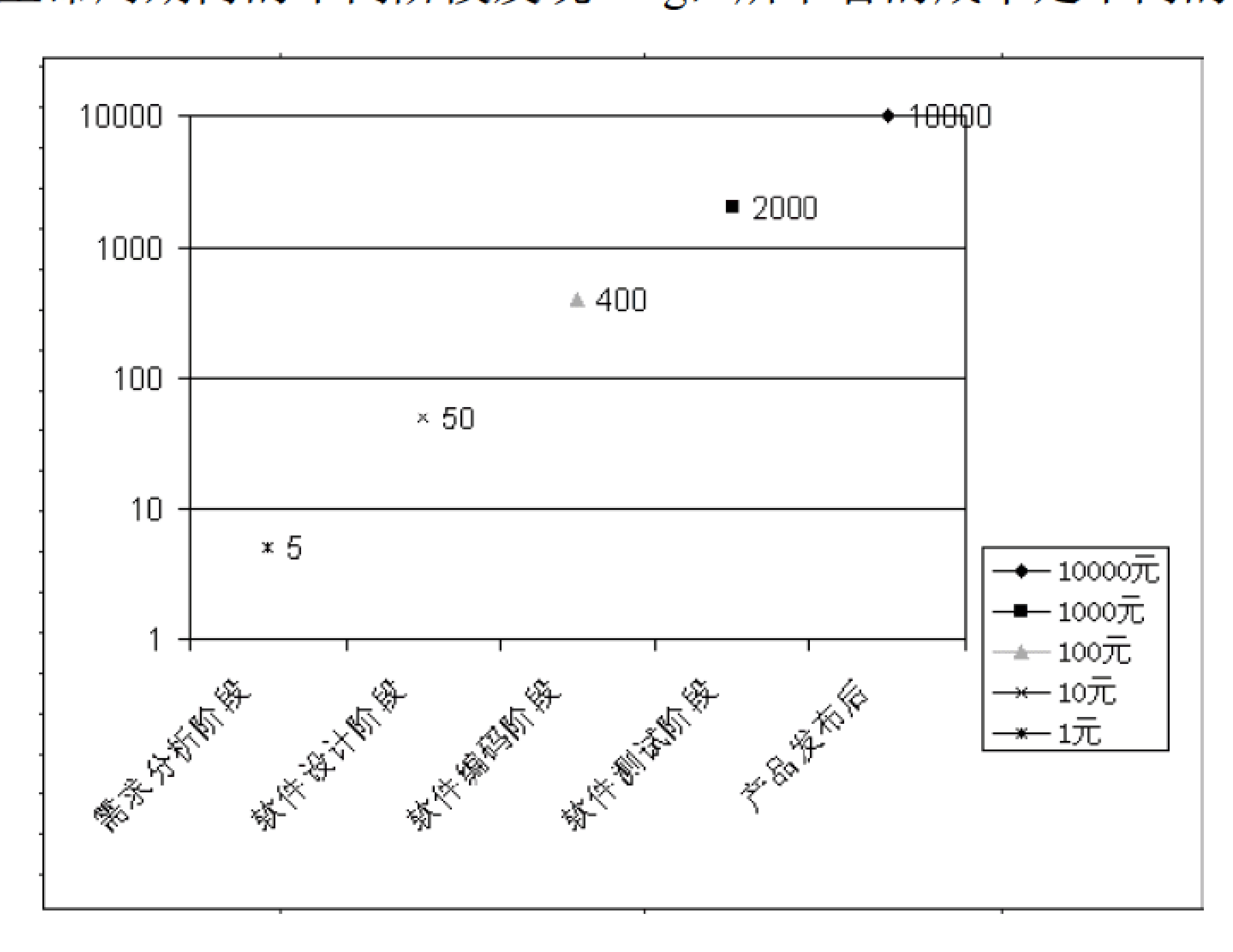


图 1-5　软件生命周期内各阶段发现与改正 Bug 所需成本示意图

从图中可以看出，在产品设计阶段发现 Bug 要比在产品维护阶段发现好得多，这是很好理解的。

- 在需求分析阶段，对于用户需求的理解停留在需求文档中，对其中理解不正确的部分只需要修改文档即可以，基本不会产生什么成本。
- 在软件设计阶段，发现的 Bug 很多都是设计思想的缺陷。由于尚未开始编码，这样的 Bug 一般需要进行深入的讨论最终获得一种正确的结论，因此改正成本也不高。
- 在软件编码阶段和测试阶段，代码通过开发人员和测试人员的努力在进行不断的完善，有关 Bug 的成本主要花费在项目内部的沟通与时间成本方面。
- 但是一旦产品发布，在软件维护阶段发现的 Bug，其修改成本会非常高昂：一是因为软件成为了系统，与开发阶段重点检查各模块功能相比更为复杂，寻找代码上的产生 Bug 根源更加困难。特别是，如果前期工作没有做好的话，甚至软件产品的结构都需要进行大修改；二是因为牵扯的部门明显增加，比如客户服务部门、产品部署部门、销售部门等都要参与，导致公司内部、公司与客户之间的沟通成

本急剧增加；第三点则是影响产品质量与公司的信誉、未来产品的销售。

【千年虫的问题】

在前几年，有一个著名的虫子把业内搅得不可开交，那就是千年虫问题，也叫做 2000 年问题。它是指在某些使用了计算机程序的智能系统（比如一般的计算机系统以及自动控制芯片等）中，由于其中的年份沿用早期的设计，只使用 2 位十进制数来表示，比如用 80 代表 1980 年，因此当系统进行（或者涉及到）跨世纪的日期处理运算（比如计算 1980 年到 2080 年之间的日期）时，就会出现错误的结果，从而引发各种各样的系统功能紊乱甚至系统崩溃。

从千年虫的实际例子中也可以看出，不考虑硬件上的限制，如果当初在设计日期表示格式的时候能够想得更长远一些，就完全可以避免这个虫子的发作，从而节省一大笔修改更新软件等的费用（据未经证实的来自美国国际资料公司调查报告表明，光是 1995 年到 1998 年，全球捉“千年虫”的开销就已经达到惊人的 1840 亿美元）。

1.3　软件测试的定义与分类

前文花费了不少文字来讲述 Bug 的定义、危害和判断原则，本节将在更广的范围内介绍软件测试的定义与分类。

1.3.1　软件测试的定义

软件测试就是利用一定的方法对软件的质量或者使用性进行判断和评估的过程。这一定义获得了较广泛的认同。

1.3.2　软件测试工程师的工作内容

软件测试是由软件测试工程师来完成的，他们的主要工作内容则是：

- 寻找软件中的 Bug，并且是越早发现越好（原因见 1.2 节）。
- 确认 Bug 的可重复性（Repro）以及 Bug 产生的步骤。
- 确认 Bug 是否被解决（Fixed）。
- 测试方法、测试计划、测试平台、测试代码、测试用例、测试文档、测试报告的确定、编写和执行。

对于小白这样刚入职的新人来说，主要工作就是前 3 项以及测试用例的编写了。在 1.4 节将讲述测试用例的知识。

1.3.3　软件测试的分类

软件测试可以有很多种分类，常见的有如下一些：

- 黑盒测试（Black box testing）；
- 白盒测试（White box testing）；

- 功能性测试（Functional testing）；
- 兼容性测试（Compatibility testing）；
- 性能测试（Performance testing）；
- 安全测试（Security testing）；
- 压力测试（Stress testing）。

虽然看起来很多很复杂，但是目前，小白所要做的工作就是先熟悉这些名词，这样在阅读众多的技术文档时，了解这些名词属于软件测试的范畴就可以了。

对于软件测试的两个核心，则有必要在第 1 章详细的介绍。这两个核心分别是测试用例和测试工程师，分别代表了软件测试的两个方面：工具和人。

1.4　软件测试的核心 I：测试用例

前文提到，测试用例代表了软件测试的工具方面，是它的核心之一。那么什么是测试用例，它又有哪些要点需要我们去掌握？

1.4.1　什么是测试用例

软件测试的核心行为就是针对要测试的软件设置测试用例。所谓测试用例，英文名为 Test Case，是一个与程序部分行为以及输入、输出相关的描述或者标识。

【测试用例的 IEEE 定义】

美国电气与电子工程师协会（IEEE，The Institute of Electrical and Electronics Engineers），它出台了一个标准的测试用例定义，即“测试用例是描述输入实际值和预期输出行为或者结果的文档，它同时也标识了测试过程结果与约束。”

在实际工作中，花费测试工程师大部分时间的，都是与测试用例相关的。

1.4.2　测试用例的几大要素

一般来说，测试用例应该清楚地描述出对被测试软件发出什么数据或者条件，以及该输入所期望的结果。在小白这样的商业网站，测试部门规定测试用例应该具备如下几个要素。

1．标识符

这一点虽然和测试用例的内容没有关系，但却是测试过程中不可缺少的。比如，小白所在的部门每周都要开一次例会，向经理或者开发部门的同事说明当前整个产品的测试状态，有时候需要特别指出某个测试用例的内容，那么用一个简单的代号来代表这一测试用例是非常适合的，这个代号一般情况下都是一个正整数，比如 1、88、437 等这样。在小白所在的公司，测试用例是存放在一个数据库中的，代号也就自然地采用了数据库系统中的标识符字段类型。如果采用其他的方式存储测试用例，可以人工指定，只要保证标识符不重复就可以了。

如图 1-6 显示了应用于真实测试场景的某测试用例文档，它实际上是一个 Office Word 文件，测试工程师（即编写者）在文件内容中手工指定了各个测试用例的标识符。

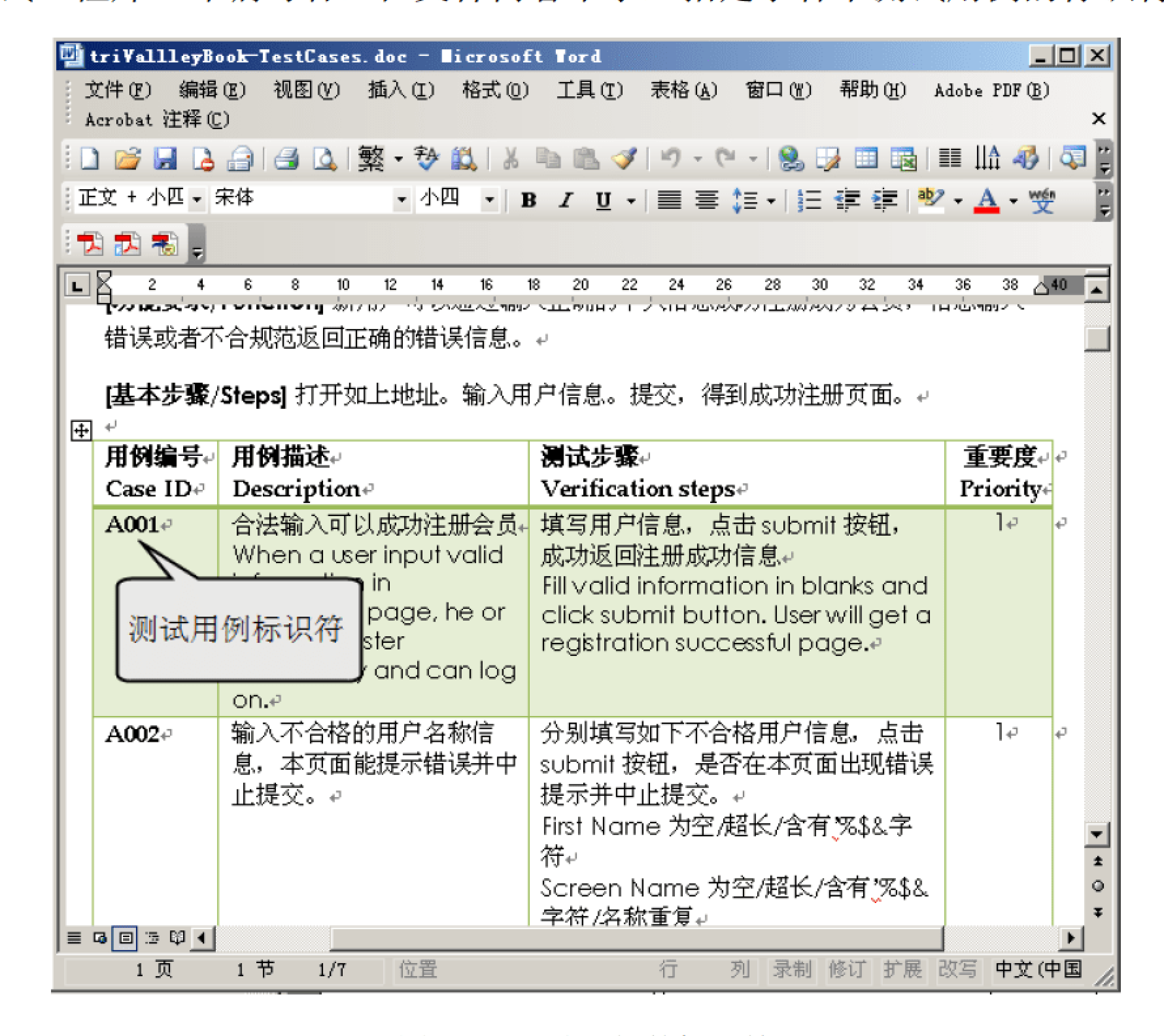


图 1-6　测试用例的标识符

2．测试的内容

测试内容可以说是测试用例最重要的部分，它一般指明了当前测试用例的运行目的，比如测试网页是否可以打开、单击按钮后是否能够显示正确的计算结果等。在很多情况下，测试内容与下个要点：输入的条件区别并不是很清楚。

3．输入的条件

输入条件可以是操作步骤，也可以是输入的数据，还可以是系统运行环境的需求（比如处于某种特别的操作系统环境内这一条件）。图 1-6 中的各个测试用例，都详细地写明了每一步骤的具体操作。

【复现步骤】

对于被测试软件而言，不同的输入条件会导致不同的输出预期，因而可能出现的 Bug 表现并不一定相同。如果重复某些输入条件，总会导致某个 Bug 的出现，那么就把这些输入条件称为 Bug 的复现步骤（Repro Step）。

4．输出的预期

该项信息描述了在当前的输入条件下，预期的输出。比如计算器程序中十进制数字的 2+3 的输出应该等于 5。若输出预期与实际结果不同，则应该考虑为 Bug 的可能性（有的

时候，输出预期也会随项目的进度而变化，因此预期与实际不同并不意味着 100%是 Bug，此时需要与项目经理等相关人员进行协商）。

5．测试环境信息

这一部分的内容描述了该测试用例所适用的环境，比如操作系统的版本，所依赖硬件软件的版本、语言等。测试环境信息有时候也可以成为输入条件或者复现步骤的一部分，比如某个按钮只有在 IE 浏览器中才会出现、某个 Bug 只在 IE 浏览器中才会产生，那么 IE 既是测试环境信息，也是输入条件和复现步骤。

6．与其他测试用例的依赖关系

在测试某些软件的时候，比如 MSN，如果登录这个测试用例都无法通过，那么剩下的发消息，发文件等测试用例也肯定继续进行（除去直接调用接口的那些测试之外），这就是一种测试用例之间的依赖关系。合理地应用测试用例之间的依赖关系，能够提高测试效率，减少无谓的测试时间浪费。

7．测试用例需要被开发、审阅、使用、维护和保存

这也是测试工作很重要的一部分。软件的说明书 Spec 可能会变化，因此测试用例需要变化，这就要求对测试用例进行增加、修改和删除。测试用例是文档，需要有固定的场所进行保存，一般是数据库或者文件。测试用例需要审阅，以达到预期的效果和更高的工作效率（重复的测试用例肯定会浪费测试工程师的时间）。

1.5　软件测试的核心 II：测试工程师

除了测试用例之外，软件测试的另一个核心，同时也更为关键，就是测试工程师了。这是因为，测试用例也是由测试工程师来编写的，受人的因素影响很大。可以说，人是决定软件成败的主要因素。本节将介绍测试工程师所必备的一些素质。

1.5.1　测试工程师与软件质量保障

有的时候我们在招聘广告上能够发现有些公司招聘测试人员的时候，列出的职位名称是软件质量保障工程师（QA，quality assurance），那么这两种称呼是否是代表同一种工作内容呢？

回答是基本一样，但有细微不同。软件测试工程师的主要职责在于发现并确认 Bug 的解决与否，而软件质量保障工程师则更进一步，在测试工程师的职责之外，还包括创建、维护为保障软件质量而确立的规范、规则与流程，比如软件配置管理（Software Configuration Management，又称 SCM 工程师）等。

【字面意义的理解】

从字面意义上来看，测试工程师主要针对软件的已有 Bug，类似体检部门；而软件质量保障工程师则不光针对已有 Bug，还对预防 Bug 的产生提出建议，类似健康顾问。当然，

在实际工作中，两者的区别并不是那么清晰的，在很多公司内部，他们所从事的工作内容是完全一致的。

1.5.2　测试工程师应该具备的素质

一个合格的测试工程师，应该具备如下专业素质：

- 具备基本的数据结构，操作系统等专业知识。这一点对于从事性能测试的人员来说更为重要。
- 具备一定的程序开发经验。掌握一到两门语言对于进行自动测试是大有益处的，另外，具有程序开发经验，也更容易理解软件 Bug 的来龙去脉。这一到两门语言可以是某些高级语言，比如 C#和 VB.net，以及一种脚本语言，比如 JavaScript、VBScript 或者 Python 等的组合。
- 软件使用经验丰富，对于软件的不正常行为敏感。这一点对于发现 Bug 是很有帮助的。

同时，测试工程师还应该具备如下的性格特征。

- 有好奇心，乐于探索软件功能，乐于尝试新的软件产品。
- 乐于探索谜题，追根溯源。对于一个 Bug，必须有追根溯源的精神，才能够发现它的特点，这个性格特征在判断 Bug 的产生原因，以及是否与其他 Bug 重复等日常的工作内容中都会展现。
- 有耐心，不轻言放弃。测试工程师在工作中经常会试图复现一个软件中的 Bug，这需要细心、耐心和坚持。
- 必须具备一定的创造性。测试工程师是无法模拟出用户使用软件的所有场景的，因此必须具备一定的创造性，通过测试更多情况下软件的不同表现，发现被测软件更多的问题。
- 具备一定的沟通和交流技巧。这一点尤为重要，测试工程师由于工作性质的要求，要给开发工程师所编写的代码找到问题。因此在平时的工作中要注意利用良好的沟通、交流技巧，使得开发工程师能够接受自己正确的观点，在保证软件质量的情况下不影响团队的合作氛围，让整个团队都体会到测试工程师和测试工作的重要性。

1.5.3　测试工程师的职业发展

软件测试工程师在我国尚属一个比较新的职业，目前专业人才相对缺乏。另外，和软件开发相比，软件测试具有进入门槛相对较低，职业生涯相对较长的特点。因此，从事软件测试的职业是具备较好的发展前途的。随着工作经验的不断积累，比较好的发展路径是：

- 初级测试工程师。进行黑盒手工功能性测试（第 2 章我们将讲解什么是黑盒测试、功能性测试）或者本地化测试等，一般是从学校毕业后参加工作的 2～3 年，正处于学习或者进一步熟悉、巩固测试基本知识与方法的阶段。
- 中级测试工程师。进行测试计划的编写，测试工具的开发，各种测试的实施等。

处于这一阶段的人员，相比于初级测试工程师工作了更长的一段时间，掌握了测试的基本理论和知识，对软件开发流程有了比较深入的了解，同时对某项软件开发技术有较深入研究的程度。

- 高级测试工程师或者测试经理。经历了初级和中级测试工程师的磨炼，从头至尾参与了一个产品或者多个产品的生命周期，在其他方面能力也具备的情况下，就可以从事这样的职位了。工作职责一般是统筹软件的测试计划，决定测试方法，确立、实现测试框架，管理控制测试进度等。
- 开发人员。实际上，从测试人员转为开发人员的比例不算小，因为测试人员对产品比较熟悉，如果自身的开发能力较强，往往具备纯开发人员所不具备的测试知识和用户视角，因此编写的代码质量更高。

以上是软件测试工程师的发展路线。在公司的选择上，一般来讲，大型或者国外的企业往往更加重视测试，所以软件测试工程师更多出现在这样的企业中，导致其平均薪资水平相对比较高。对于想从事初中级测试工程师的读者来说，一般可以考虑如下 4 类公司。

- 国内的 IT 相关企业。这些企业大致有各软件开发公司，通信、电子、游戏公司等。
- 外资企业在国内的研发中心。这类公司的产品开发人员相对更集中在公司总部，同时为了节省成本，测试工程师则较多地安排在国内。
- 为外资企业做软件外包的国内企业。微软、Google、IBM 等很多在华的跨国企业都会聘用这些企业的员工进行外包测试工程师的工作。
- 大型网站。中小型网站往往没有专职的测试人员；大型网站由于架构较复杂，分工相对更为专一，会有专职的测试工程师队伍以保证质量。

1.6　本章小结

本章通过刚刚作为软件测试工程师新人的小白短短几天的经历，对软件测试的基础知识进行了简要的介绍。

在本章的开头，给出了公认的软件定义。所谓软件，就是计算机系统中的程序和相关文件或文档的总称。

随着软件越来越复杂，软件危机逐渐出现，人们为了从软件开发的初期就避免出现这样的问题，研究总结出若干软件生命周期的模型，主要有如下 4 种：

- Big-Bang 大爆炸模型；
- Code and Fix 边做边改模型；
- Waterfall 瀑布模型；
- Spiral 螺旋模型。

通过在工作中应用这些模型，很好的进行了软件项目管理。

但是，软件中的 Bug 依然不可避免。所谓 Bug 就是软件的错误或者偏差。在软件开发的过程中，越早发现 Bug 就越能节约软件开发的成本。

编写测试用例，利用人工或者自动的方法寻找被测试软件 Bug 并确认修复，是软件测试工程师在工作中要完成的职责，加上有效的预防 Bug 产生的规范，就可以在很大程度上保障软件的质量。

第 2 章　测试方法与过程

第 1 章已经对测试的两个重要组成部分（软件测试工程师和测试用例）进行了简单的介绍。本章讲解软件测试工程师编写和运用测试用例的一般方法，同时，还要介绍一些测试工作周边的相关知识，使小白这个新人尽快进入测试的氛围，并为第 3 章正式学习 Web 测试做好准备。

2.1　测试的主要方法与分类

俗语说的好，“大事化小，小事化了”。在进入一个全新的领域开始学习之前，把它划分为小块来熟悉是一个不错的方法。我们学习软件测试同样也要采取这样的办法。根据测试时行为和应用场景的不同，可以把软件测试划为很多种分类，比如：

- 黑盒测试和白盒测试；
- 功能测试；
- 压力测试；
- 代码覆盖测试；
- 本地化测试；
- 回归测试。

本节将对上面所列的这些测试分类逐一进行简单介绍，相信小白在不久的将来就会经常遇到它们。

2.1.1　白与黑

其实，这里所讲的白与黑是指测试业内中出现的两个术语：白盒测试与黑盒测试。特别地，这两个词语在一些公司的招聘广告中也会经常见到。

相对于白盒测试，更多的测试工程师在从事黑盒测试的工作，那么就先从黑盒测试说起。

1．黑盒测试

所谓黑盒，就是形容测试工程师对软件内部如何实现不了解的那种状态。黑盒测试是以外部的视角来观察软件的。黑盒测试就是软件测试工程师将有效或者无效的测试用例输入到被测试软件当中，观察软件输出的这么一种行为，英文学名叫做 Black-box testing。

图 2-1 是黑盒测试示意图，从图中可以明白，测试工程师对软件内部的具体实现（代码中有哪些方法，有什么类等）不需要了解，只关心输入和输出。

这和我们在电脑大卖场购买主机有点类似，我们可以不需要有电子工程师、硬件工程师那样的知识，也不需要知道主机内主板各总线的布线方式，电路的排布等。作为消费者，只要输入合适的电压，电脑就应该能够输出图像；只要输入合适的数据，电脑就应该能够计算出正确的结果。如果这两项出现了问题，那么我们不用拆开主机，就能够宣布这台电脑出了问题。

2. 白盒测试

白盒测试则与黑盒测试相反，测试工程师需要了解测软件中的结构，因此它需要测试人员有较高的编程技巧。基于这种特点，白盒测试也叫做结构化测试、玻璃盒测试，英文学名分别为 White-box testing、Structural testing 和 Glass box testing。

我们同样利用电脑主机来说明白盒测试。在主机卖给消费者、或者使用中出现了问题时，都需要对其进行检测。在这样的情况下，专业人员会拆开主机箱，一个零件一个零件地排除问题，还会利用一些测试卡，通过读取上面的数据来判断具体的错误类型。而且，每个厂家的主板都有自己的特点，因此有自己的维修队伍。这样的情况就很类似白盒测试。

图 2-2 是白盒测试的示意图。其中列举了两条软件执行的路径 A 和 B，对于进行白盒测试的工程师来说，白盒中的路径是需要他们关心的。

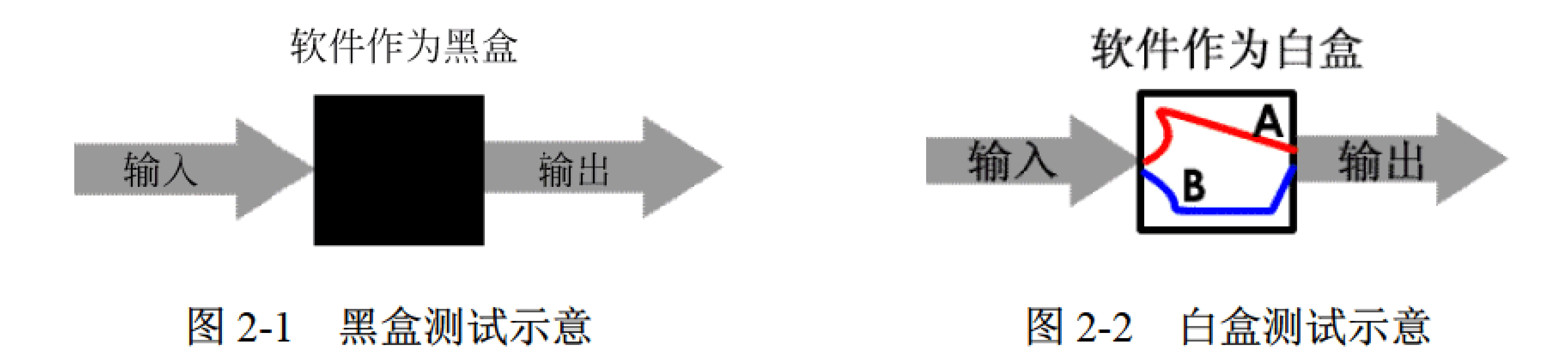


图 2-1　黑盒测试示意　　　　图 2-2　白盒测试示意

联系到本书的主题，针对 Web 网站的性能测试，可以用黑盒的方法来确认网站是否具有性能问题，利用白盒的方法来确认哪里产生了性能问题。

2.1.2　黑盒与白盒测试的比较

黑盒测试和白盒测试的优缺点比较如表 2-1 所示。

表 2-1　黑盒测试与白盒测试的优缺点

类　别	优　点	缺　点
黑盒测试	不需要了解软件实现细节 软件内部实现机制更改时，一般不必修改测试用例 实现相对简单 以用户使用角度出发	无法保证软件代码内各主要路径都被覆盖到，容易导致测试不很完全
白盒测试	针对软件代码各路径进行测试，相对易于调试，容易发现 Bug 产生的原因	对测试人员的编程能力要求高。 软件实现代码改变，测试用例一般也需要改变

前文提到过，大部分软件测试工程师利用的都是黑盒测试方法。因此，掌握它的知识在初级测试阶段就相对比较重要。在 2.1.3 节中我们将重点介绍黑盒测试。

有关白盒测试的方法，由于涉及更多的编程知识，需要在实际工作中不断地磨炼和提高，在本书后面的章节中会偶尔提及。

在实际工作中还有一种测试方法，称为灰盒测试。顾名思义，它是把白盒测试和黑盒测试结合起来的方法，但是掌握这样的方法，需要掌握黑盒测试和白盒测试的知识，因此在这里就不单独讲解了。

2.1.3　黑盒测试方法简介

前文提到，黑盒测试是把被测试软件看作一个黑盒子。利用黑盒测试一般应用于测试软件的功能，不能测试软件产品的内部结构和处理过程。

1. 黑盒测试的侧重点

黑盒测试注重于测试软件的功能性需求，即黑盒测试使软件工程师派生出执行程序所有功能需求的输入条件。黑盒测试并不是白盒测试的替代品，而是用于辅助白盒测试发现其他类型的错误。

2. 黑盒测试能发现的错误

黑盒测试试图发现以下类型的错误。

- 功能错误或遗漏：比如单击“退出”按钮后软件没有退出这样的错误。
- 界面错误；比如软件界面上有些按钮只显示了一半这样的错误。
- 数据结构或外部数据库访问错误：比如在数据量大时、网络异常时软件不可用的情况。
- 性能错误：比如长时间运行软件导致死机的情况。
- 初始化和终止错误：比如软件在某种条件下无法启动的情况。

3. 黑盒测试的测试用例设计原则

测试用例的设计原则体现了测试方法。一般来说，判断测试用例是否设计得好，有如下两个标准。

- 在测试覆盖率（也就是测试过的代码所占全部软件代码的比例）相同的时候，测试用例要尽可能的少。测试用例少，则运行测试用例所花的时间和其他成本就比较节省。
- 现有的测试用例要达到越高越好的测试覆盖率。只有测试覆盖率提高了，我们才能对软件的质量有更明确的判断和信心。

对于黑盒测试来说，测试用例设计方法主要有如下几种。

- 等价类划分方法：关注输入的值域。
- 边界值分析方法：关注不同值域之间的变化点。
- 错误推测方法：关注代码变动的点、关注历史上 Bug 较多的点。
- 因果图方法：关注被测试软件逻辑上的因果关系。
- 判定表驱动分析方法：关注被测试软件中的条件选择顺序。
- 正交实验设计方法。

- 功能图分析方法。

从下节开始，小白将陆续学习到一些常用的黑盒测试方法。

2.2　等价类划分方法

等价类划分是必须掌握的一个重要的黑盒测试基本方法。使用等价类划分，是为了提高编写测试用例的效率，完善测试用例对软件可能输入数值范围的覆盖率。

2.2.1　什么是等价类划分

等价类划分，英文名为 Equivalence Partition，是把所有可能的输入数据，即程序的输入域划分成若干个部分，然后再从每一个部分当中选取一个或者少数具有代表性的数据作为测试用例的输入。概念理解起来会比较枯燥，小白给我们举出了一个例子。

如图 2-3 是国内某著名旅行网站的首页。在首页上有两个文本框，分别代表酒店的入住日期和离店日期。假设小白要对这两个日期输入框进行测试，他首先想到的是输入一个正常的月份，比如 3 月，看网页能否正常提交；然后输入一个不存在的月份，比如 0 月，查看网页的结果。这是很多人都会想到的测试数据，那么，输入的月份数据 3 提交后正确是否能够代表所有的有效月份都能提交正确呢？这就需要理解等价类的划分了。

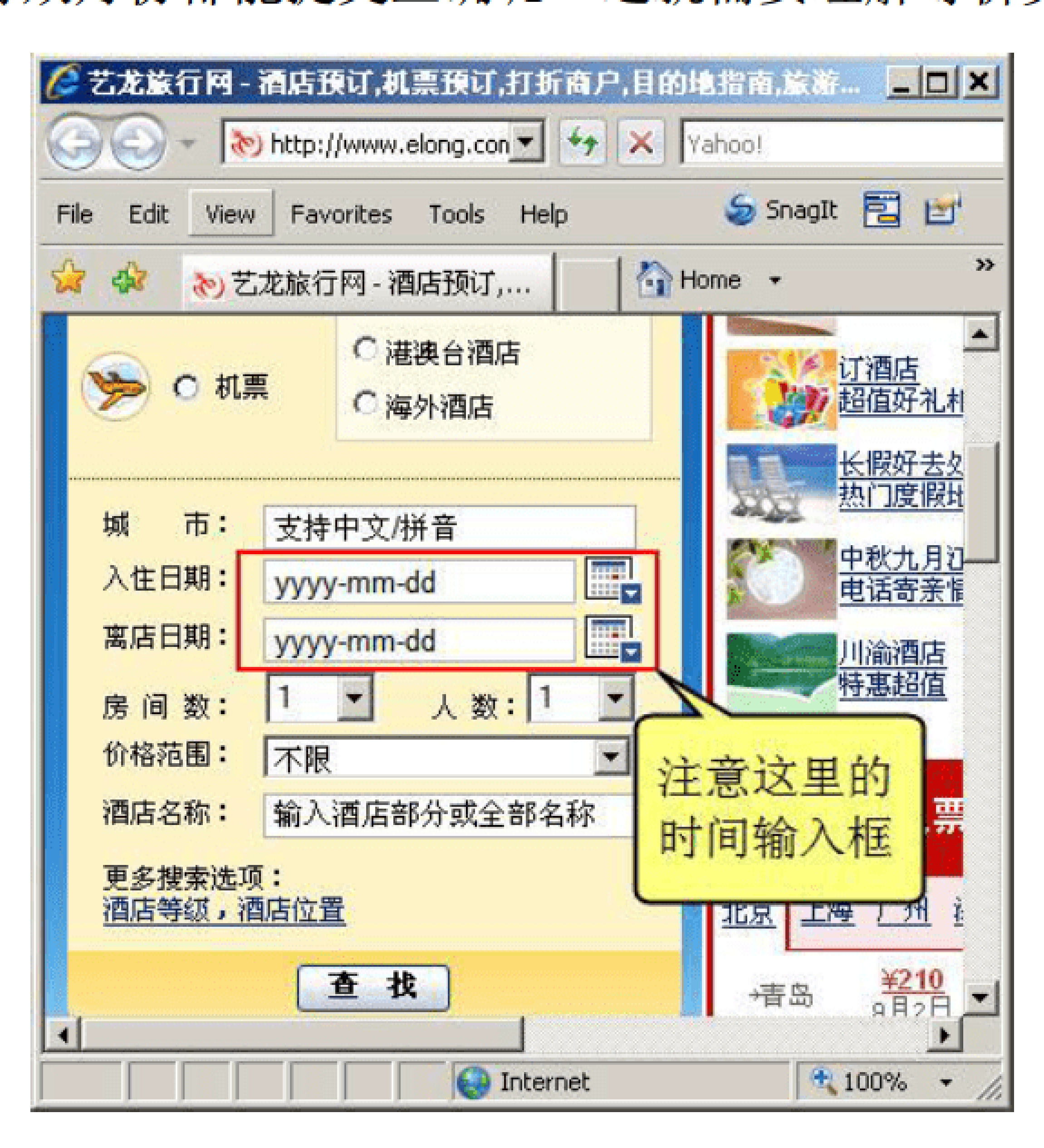


图 2-3　对某旅行网站的日期输入框进行测试

【等价类】

等价类是指某个输入数据范围的子集合。在该子集合中，被测试的软件程序对于其中任意输入数据所产生的行为都是一致的。因此可以合理地假定，输入某等价类的某个代表数据值进行测试，就等于对这一等价类内其他数据值的测试。因此，通过把全部可能的输入数据进行合理划分，在产生的每一个等价类中取一个数据作为测试的输入条件，就可以

用少量具有代表性的测试数据，取得与输入很多数据时一致的测试结果。

根据所选取输入数据的有效性，等价类划分可有两种不同的情况。

- 有效等价类：指对于软件的规格说明来说是合理的、有意义的输入数据所构成的集合。利用有效等价类可检验程序是否实现了规格说明中所规定的功能和性能。
- 无效等价类：无效等价类则正好相反，利用它可以验证软件是否实现了规格说明中所规定的意外处理。

下面举例说明等价类划分的方法。

在小白测试的这个网页中，输入的数据要求是时间。为了简单一些，我们只关心月份。对于月份来说，1～12 这些数字是符合现实情况的。根据这个生活常识，可以把所有用户可能输入的月份数字划分成几个部分，如图 2-4 所示。

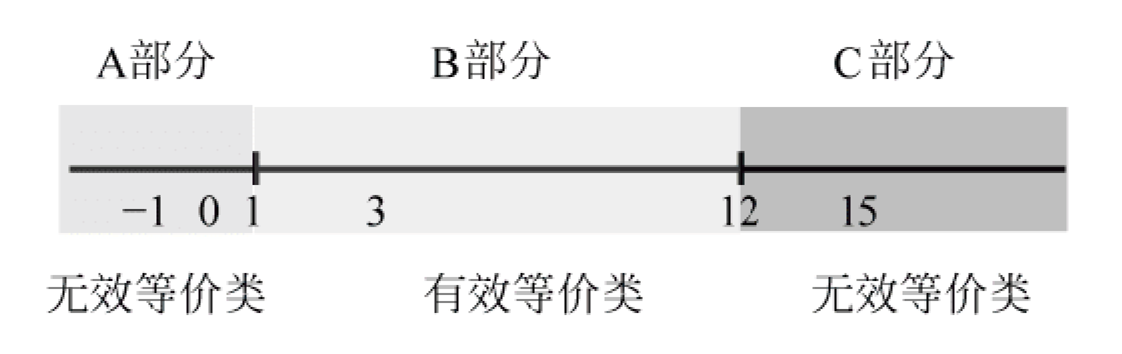


图 2-4　月份日期的等价类划分

在图 2-4 中，如果输入任何一个小于 1 的月份（即 A 部分）都会造成网页同样的行为，那么这些数字就构成了一个等价类，同时，由于它们的数字是无效月份，因此我们把这样的等价类称为无效等价类。

对于 1～12 之间的 B 部分来说，如果输入其中任何一个数字所产生的网页行为一致，那么这些数字就构成了另一个等价类，这些数字符合常识，因此月份有效，我们把这样的等价类称为有效等价类。

对于大于 12 的 C 部分来说，同样构成了一个无效的等价类。

2.2.2　等价类划分的标准

等价类划分有个标准，即软件行为的规格说明。对于图 2-4 所举的例子来说，如果软件 Spec 中指明输入 A 部分或者 C 部分的数字软件的行为都一致，或者实际发生的情况如此，则可以将 A 部分和 C 部分合并为一个无效的等价类。

等价类划分很好地实现了前文所列出的好的测试用例设计要求。

通过选取代表性的数据，导致输入数据减少，从而测试用例数量减少，而测试的覆盖率并没有减少。

从这一句话也可以看出，等价类的确定至关重要。如果选取的等价类不准确，必然导致有部分场景没有测试完全。

2.2.3　划分等价类的方法

下面给出 6 条确定等价类的原则：

- 在输入条件规定了取值范围或值的个数的情况下，则可以确立一个有效等价类和两个无效等价类。
- 在输入条件规定了输入值的集合或者规定了“必须如何”的条件下，可确立一个有效等价类和一个无效等价类。
- 在输入条件是一个布尔量的情况下，可确定一个有效等价类和一个无效等价类。
- 在规定了输入数据的一组值（假定 n 个），并且程序要对每一个输入值分别处理

的情况下，可确立 *n* 个有效等价类和一个无效等价类。

- 在规定了输入数据必须遵守的规则的情况下，可确立一个有效等价类（符合规则）和若干个无效等价类（从不同角度违反规则）。
- 在确知已划分的等价类中各元素在程序处理中的不同方式的情况下，则应再将该等价类进一步地划分为更小的等价类。

2.2.4　利用等价类划分设计测试用例

在确立了等价类后，可建立类似表 2-2 的等价类表，将所有划分出的等价类列出，以备设计测试用例时全面考虑。

表 2-2　关于月份测试的等价类表

输入条件	有效等价类	无效等价类
月份	1～12 的整数	小于 1 或者大于 12 的整数
		字母
		其他符号

然后从划分出的等价类表中按照如下 3 个步骤开始设计测试用例。

（1）为每一个等价类规定一个唯一的编号，这样做是为了标识不同的测试用例。

（2）设计一个新的测试用例，使其尽可能多地覆盖尚未覆盖的有效等价类。根据未覆盖的有效等价类数量重复这一步，直到所有的有效等价类都由测试用例覆盖为止。

（3）设计一个新的测试用例，使其仅覆盖一个尚未被覆盖的无效等价类。根据未覆盖的无效等价类数量重复这一步，直到所有的无效等价类都被覆盖为止。

通过上述的这 3 个步骤，就可以达到前文所说的好的测试用例设计目标。

在实际使用中，有效等价类和无效等价类的连接处经常是产生 Bug 的多发地带，输入这样的数据对于判断软件的质量很重要，这样就用到了边界值分析法。等价类划分和边界值分析法基本上都是同时使用的。

2.3　边界值分析法

边界值分析方法是对等价类划分方法的补充。所谓边界值，就是两个相邻等价类之间的那个数值。利用边界值分析法，对某些“敏感”区域进行特殊测试，有可能在很短的时间内就发现较多 Bug，有利于提高测试效率。

2.3.1　边界值分析法的数据选取原则

由于软件代码内部一般包含大量的判断，而这些判断导致了不同输入数值所产生的软件行为的不同，因此，大量的软件 Bug 发生在这样的判断条件上，导致对这些判断条件的验证具有非常重要的意义。根据等价类的定义，等价类一般来说正是由于满足判断条件的输入数值所划分的。针对各种等价类的边界情况设计测试用例，可以更有效率地查出更多

的 Bug。

【确定边界非常重要】

使用边界值分析方法设计测试用例，首先应该确定准确的边界情况。前文已经提到，输入和输出数据中等价类的各个边界，就是应着重测试的边界情况。应当选取正好等于、刚刚大于、或者刚刚小于边界值的输入数据进行测试，而不必选取等价类中远离边界值的数据。

2.3.2　根据边界值分析法设计测试用例的原则

由于实际的软件往往较复杂，边界值不是很容易发现，因此有必要遵循一些固定的“模式”来创建测试用例，即如下的一些原则：

- 如果输入条件规定了值的范围，则应取刚达到这个范围的边界值，以及刚刚超越这个范围的边界值作为测试输入数据。
- 如果输入条件规定了值的个数，则用最大个数、最小个数、比最小个数少 1、比最大个数多 1 的数作为测试数据。
- 根据规格说明的每个输出条件，使用前面的第一个原则。
- 根据规格说明的每个输出条件，使用前面的第二个原则。
- 如果程序的规格说明给出的输入域或输出域是有序集合，则应选取集合的第一个元素和最后一个元素作为测试用例。
- 如果程序中使用了一个内部数据结构，则应当选择这个内部数据结构的边界上值作为测试用例。
- 分析规格说明，找出其他可能的边界条件。

【实战演习】

假设需要测试取款机内运行的软件，要求每天最多只能取出 10000 元的现金，每一次操作钱箱最多吐出 2500 元的现金。如果输入的数据不符合规范，将提示出错的信息，同时中止操作，返回选择金额的界面。那么，我们根据如上的原则，可以选取 2550、2501、2600 这几个数字来输入，测试软件的输出；还可以选取 2000 元 4 次，2100 元 1 次这样的操作组合作为输入，从而获得软件的实际运行情况，发现问题。

2.4　判定表方法

前面介绍的等价类划分方法和边界值分析方法，都是着重考虑输入条件，但未考虑输入条件之间的联系，相互组合等。考虑输入条件之间的相互组合，可能会产生一些新的情况。但要检查输入条件的组合不是一件容易的事情，即使把所有输入条件划分成等价类，它们之间的组合情况也相当多。因此必须考虑采用一种适合描述对于多种条件的组合，相应产生多个动作的形式来考虑设计测试用例。这就需要利用决策树方法，或者叫做因果图（逻辑模型）。

判定表方法得名于因果图最终生成的表格名字。它比较适合于检查程序包含条件判断时，不同输入条件的各种组合情况。判定方法有 5 个步骤，主要部分是如下两个。

- 生成因果图。

- ❑ 根据因果图形成判定表，确定测试用例。

2.4.1　判定表生成测试用例的 5 个步骤

与其他测试用例生成方法类似，判定表方法也有自己的一套原则可供遵循。

（1）分析软件规格说明书中，哪些是原因（即输入条件或输入条件的等价类），哪些是随之产生的结果（也就是输出），并给每个原因和结果赋予一个标识符。

（2）分析软件规格说明书中的逻辑关系。发现并记录各个原因与结果之间、原因与原因之间对应的关系，根据前述这些关系，画出因果图。

（3）由于某些限制，比如说软件说明书中的语法或者是当前的认识限制，有些原因与原因之间、原因与结果之间的组合情况不可能明确出现。对于这些关系，在因果图上用一些记号表明。

（4）将因果图转换为判定表。

（5）把判定表的每一列拿出来作为依据，设计测试用例。

在最后一步中，要争取由判定表生成的测试用例包括了所有输入数据的取 TRUE 与取 FALSE 的情况，构成的测试用例数目达到最少，且测试用例数目随输入数据数目的增加而线性地增加。

2.4.2　判定表的结构

本节简单介绍判定表的结构，不过在这之前，有必要了解判定表是什么。

【判定表是什么】

判定表（Decision Table）是分析和表达多逻辑条件下执行不同操作的情况下的工具。在程序设计发展的初期，由于判定表能够把复杂的逻辑关系和多种条件组合的情况表达得既具体又明确。它就被很多专业人士采用，作为程序编写的辅助工具了。

判定表通常由 5 个部分组成。

- ❑ 条件桩（Condition Stub）：列出了问题的所有条件。通常认为列出的条件次序无关紧要。
- ❑ 动作桩（Action Stub）：列出了问题规定可能采取的操作。这些操作的排列顺序没有约束。
- ❑ 条件项（Condition Entry）：列出针对它左列条件的取值。在所有可能情况下的真假值。
- ❑ 动作项（Action Entry）：列出在条件项的各种取值情况下应该采取的动作。
- ❑ 规则：任何一个条件组合的特定取值及其相应要执行的操作。在判定表中贯穿条件项和动作项的一列就是一条规则。显然，判定表中列出多少组条件取值，也就有多少条规则，实际上就是条件项和动作项共有多少列。

2.4.3　判定表的建立步骤

建立一个合格的判定表，有如下 5 个步骤。

（1）确定规则的个数。假如有两个条件，每个条件有两个取值（0,1），则通过排列组合，有 2×2=4 种规则，即 00、01、10、11　4 项。

（2）列出所有的条件桩和动作桩。

（3）生成判定表的记录，填入条件项。

（4）填入动作项。

（5）简化过程，合并相似规则以确保测试用例较少。

通过这样的过程，可以保证该方法产生的测试用例不会有所遗漏。

2.5　其他黑盒测试方法

编写测试用例还有其他不少方法，而本节主要介绍错误推测法。

【错误推测法】

错误推测法是基于经验和直觉推测程序中所有可能存在的各种错误，从而有针对性地设计测试用例的方法。

错误推测法的基本思想主要依据如下原理：

在 Bug 出现次数多的地方出现下一个 Bug 的几率相对更大。

这是在软件测试领域中很著名的原理。由于需要依靠经验和已有 Bug 出现的频度数据，错误推测法主要使用在项目的中后期，首先列举出程序中所有可能有的错误，判断它们的特点，找到容易发生错误的危险地带和特殊情况，根据结果选择或者创建新的测试用例。

例如：可以根据在单元测试时曾列出的许多在模块中常见的错误、上个版本产品测试中曾经发现的错误等来决定当前的测试用例，这些都体现了经验的总结。

【错误推测与经验积累】

错误推测首先需要建立在对错误的较多了解之上。测试工程师在工作之余可以统计一下被测试软件在哪些方面 Bug 最多，而这些 Bug 的性质又具备什么样的特点。另外，对于新增加的代码，有理由认为带来的 Bug 更多。测试工程师可以对这些信息进行统计，并加以记录和保留，这样，在今后测试完成类似功能的软件时可以利用它。

当然，还有其他一些编写测试用例的方法，感兴趣的读者可以参考相关的书籍。

2.6　测试分类简介 I：性能与代码覆盖

本节简要介绍工作中可能遇到的各种测试分类，使读者能够大致了解。需要指出的是，这些测试的分类标准与黑盒、白盒测试是相互独立的（或者叫“正交”的），好比描述一个朋友，可以从身高方面，也可以从爱好方面，更可以从性格方面来着手。

2.6.1　性能测试与压力测试

本书主要关注的性能测试和压力测试也是很重要的一部分，将在后面的章节中详细讲述。本节将简单提出名词，使大家留有一般的印象。

如果从事性能测试的工作，在文档中经常可以看到如下一些类似名词。

- 负载测试（Load Test）；
- 压力测试（Stress Test）；
- 容量测试（Capability Test）；
- 性能测试（Performance Test）。

那么它们彼此之间有什么联系和区别呢？

其实性能测试是一个范围较广的领域，它具体包括负载测试、压力测试和容量测试。负载测试是为了检验软件在给定负载下（比如网络带宽在一定数值下）是否能达到软件说明书中预期的性能指标；压力测试是通过不断向被测软件施加“压力”（比如同时有 1 万人访问网页，同时有 10 万人访问网页等情况），测试系统在超过什么样的压力下性能表现会发生大的变化，从中发现瓶颈并加以改进；容量测试则一般针对某数据库应用类软件，通过在数据库中不断增加数据记录的方法对整个系统的最大容量进行测试。

【性能测试的基本过程】

在性能测试的过程中，根据被测试软件的不同类型与测试的不同目标，记录的数据也不同。例如，对于以调优为目的的性能测试，可能需要重点关注测试过程中各种可能的性能制约点（例如磁盘 IO、网络拥塞状况、服务器内存使用情况、数据库使用情况等），通过对参数调整后的系统进行反复测试来找到制约性能的因素；而一个以验证为目的的性能测试可能会重点关注是否能达到性能指标要求，重点集中在用户体验上。

【性能测试所使用的工具】

从一般的应用场景来说，性能测试由于需要模拟用户的并发等操作，人工无法很好地实现（设想奥运会百米赛跑的场景，很少有人听到枪声的反应时间是一样的，对于单击某个按钮也是如此），需要测试工具的良好支持才能进行更准确的性能测试。

业内常见的性能测试工具，主要有 Mercury 公司（已经被 HP 公司收购）的 Load Runner、IBM 公司 Rational 系列中的 Load Tester 等，还有很多的开源测试工具，我们将在今后的章节中详细介绍。当然，在读者的经验和开发能力提高之后，完全可以自行开发适应本地情况的性能测试工具。

2.6.2　行路难：代码覆盖

代码覆盖，英文名称为 Code Coverage，是考察软件代码已经被测试程度的一类测试。由于这种测试涉及软件实现的代码本身，因此它属于白盒测试的一部分。

大约在 1963 年前后，某些程序开发人员首先想到了代码覆盖的意义，认为它测试的结果之一，即代码覆盖百分比，或者说代码覆盖率可以代表了软件被测试的比例。我们知道，软件的代码中有很多的判断、分支和循环，不同执行顺序所产生的软件行为是不一样的，当把所有的这些路径都验证一遍的情况下，代码覆盖率是 100%。

【代码覆盖与游戏】

举一个例子来说明。有些走迷宫类的角色扮演游戏会在游戏完成时给玩家一个百分比，比如 75%，提醒玩家还有 25%的情节（路径）没有经历到，如果游戏有意思的话，玩家会从头开始，在途中颠覆之前的选择，这样就很可能会面临一个全新的结局，提高游戏的好玩程度。代码覆盖有点类似游戏中的这种情节遍历，遍历得越多，说明软件代码被测试过的比例越大，对软件质量、测试结果的信心也相对会增强。

代码覆盖测试同样会给出一个百分比作为指标，但依据考察的标准不同，主要可以分为几种代码覆盖率。

1．区分几种代码覆盖率

在实际工作中，代码覆盖率主要有以下几种：

- 函数覆盖率：在代码所有的函数中，测试时执行的函数占多大比例？
- 语句覆盖率：在代码所有的行中，测试时执行的行占多大比例？
- 条件覆盖率或者叫做分支覆盖率：在代码所有的判断中，测试时执行的判断占多大比例？
- 路径覆盖率：在代码所有可能的路径中，测试时执行的路径占多大比例？

从上面的分类中可以发现，不同代码覆盖率选取的标准不同，而且可能互相包含，比如，函数覆盖率是在函数级别上进行覆盖测试，而每一个函数内部则会有很多的代码行。对于要求严格的软件来说，由于测试时间、人手和测试工具的限制，针对不同代码覆盖率，标准也不同：函数覆盖率要达到 100%；条件覆盖率尽可能达到 100%；而语句覆盖率等超过 70%已经比较合适了。

2．不可完成的任务：100%路径覆盖率

在现实情况之中，出现 100%的路径覆盖率几乎是不可能的（除非某些很简单的软件）。这是因为代码中的选择判断语句和循环语句将使得路径的数量空前增长。而且，有些路径在一般情况下是根本走不到的。

假设一个条件语句 If，那么由于这条语句就有了两种选择，真或者假。这样，如果条件语句有 N 个判断，就会形成 2^n 种选择，如果再加上循环，就会使得路径的数量空前增长。

【实际工作中的代码覆盖】

实际工作中常用的是语句覆盖率和路径覆盖率。由于代码覆盖需要较多的编程知识和程序调试技巧，一般依赖于专门工具来进行，比如 ，由软件开发工程师在做单元测试、测试工程师在做系统测试的时候使用，两者所得到的百分比如果不断地提高，都有助于对软件产品质量信心的建立。

2.7　测试分类简介 II：本地化与国际化

世界是平的。在全球化的今天，很多大中型、甚至很小的软件都有各语言的版本，这使得更多的人能够享受到软件给人带来的便利和工作效率的提高。

小白所在的公司也同样是这样：他们的网站有多个语言版本，面对来自世界各地的浏览者。于是，小白也经常接触到本地化和全球化这两个名词。

2.7.1　国际化与 i18n

国际化这个名词翻译自英文单词 Internationalization。因为这个单词确实稍微长了一点，所以人们就找了一个简短的方式来称呼它：i18n。具体数一下单词开头（字母 i）和结尾（字

母 n）之间所有字母的数量，正好是 18 个，这就是 i18n 这个缩写的来历，有点类似于我们在某些文学作品中碰到的“以下作者省略 500 字”这种方法。另外，国际化也有称为全球化的，由英文单词 Globalization 翻译而来。

【国际化的含义】

一般说来，国际化的含义是指把原来为某种初始语言设计的计算机系统或应用软件改写为同时支持多种语言和文化习俗的过程。通常在软件开发的初期，一般的编程语言、编译、开发都是只支持英文的，为了适应更广泛的、不同国家和民族的语言以及文化习俗，软件有必要在设计结构和机制上支持多语言的扩展特性，这一过程称为（将软件）国际化。

从前面的叙述我们可以了解到国际化这个词用在软件开发和测试方面比较多，我这里举出一个例子来具体说一下国际化的必要性。

小白所在的公司购买了微软公司开发的 Office 2007 系列软件，那么其中英文版的 Word 2007 能否安装在同事的中文 XP 系统上呢？如果能够正常安装、正常使用，那么就可以说英文 Word 2007 对于中文 Windows XP 操作系统的国际化支持很好。可见，国际化对于软件还是非常重要的。

2.7.2　本地化与 Localization

和国际化的 i18n 类似，本地化也有个简称 L10n，这个由来是一样的。本地化则与国际化、或者说全球化不同，这种测试是验证显示界面为当地语言的软件在该语言版本的操作系统下能否正常工作，并且符合当地规则和习惯。图 2-5 来自制作北京烤鸭的著名老字号——全聚德的网站，注意全聚德招牌中 3 个汉字的排列方式。

在我国古代，横向的文字都是从右往左书写的。在当今世界上，还依然存在这样的书写方式。比如阿拉伯国家还有我们的维吾尔族等，他们的语言都是从右到左进行书写的。因此，如果你的网页要提供给这些地区这些民族的访问者来浏览的话，就一定要注意页面显示文字的方向了。图 2-6 来自一家页面主题为阿拉伯艺术的网站，在这个网站上，阿拉伯语和英语混杂使用，文字有时候从左到右进行阅读，有时候又从右到左阅读。从图中可以看出，在这样的应用场合，网页代码中实现按照需要制定文字排列方向还是很有必要的。



图 2-5　全聚德牌匾上文字的排列方式



图 2-6　某阿拉伯网站上的文字排列方式

本地化测试结果的好与坏，会影响 Web 应用的可用性，以及用户体验是否良好。

2.7.3 国际化测试与本地化测试的区别

这两个名词是初入测试行业的工程师很容易混淆的概念之一。它们的区别表面上看起来很难讲述清楚，其实国际化测试与本地化测试的区别可以用如下很简单的两句话代替。

- ❑ 国际化测试面向英文软件或者网页，在其他语言版本的操作系统或者浏览器中的表现。
- ❑ 本地化测试面向本地语言版本软件或者网页，在本地语言版本的操作系统或者浏览器中的表现。

2.7.4 国际化、本地化测试的具体内容

这两项测试并不是以测试软件的全部功能为主要目的，具体内容可以分为下面几个部分。

- ❑ 发现、验证翻译错误。
- ❑ 发现与国际化或者本地化相关的功能错误。
- ❑ 发现、验证软件界面错误。
- ❑ 发现、验证用户习惯错误。

对于第 1 种错误，常见的例子有菜单翻译的不准确等。笔者曾经看到一款机器翻译出来的软件菜单，Visual Basic 被翻译成“可视的基本”，这是不符合可用性要求的。

对于第 2 种错误，常见的例子有双字节的问题。东亚大部分国家的字符编码是双字节的，因此程序如果不支持的话，会导致软件行为异常。

对于第 3 种错误，常见的例子有软件由于各语言下表达相同意思的文字长度不同，导致界面布局混乱、出现错误等。

对于第 4 种错误，常见的例子有字符排列顺序不符合习惯，货币、纪年等不符合习惯等。比如，泰国的纪年以佛历计算，而我们一般都以公元来计算，在年份转换时可能会产生一些问题。

当然，以上几种错误有的时候是互相影响的，很难区分。这些错误产生的原因不外乎以下两点：

- ❑ 由源程序软件编码错误引起的。
- ❑ 由软件设计开发者对国际化、本地化信息掌握不够全面引起的。

2.7.5 国际化、本地化测试的简要步骤

国际化与本地化这两种测试一般来说，都需要参照母语源程序的测试结果，以确认软件出现的 Bug 是在国际化或者本地化过程中引入的。

具体而言，进行这两种测试的步骤如下：

- ❑ 国际化测试一般对源程序软件进行源程序语言系统下的测试，再在不同语言版本的操作系统或者浏览器中进行测试。

- 本地化测试也要先对源程序软件进行源程序语言系统下的测试，然后创建本地化软件，在本地语言版本操作系统或者浏览器中再进行测试。

本书只是简单地介绍了国际化、本地化测试，实际工作中它们对性能一般不会有什么影响，因此不再赘述。感兴趣的读者可以参阅相关专业书籍。

2.8　各种测试简介 III：回归、人工与自动测试

除了前文提到的这几种测试分类之外，其他经常见到的测试还有回归测试、单元测试、人工测试与自动测试等。

2.8.1　回归测试

在软件生命周期中的任何一个阶段，只要软件发生了改变，就可能给该软件带来问题。软件的改变可能是源于发现了错误并做了修改，也有可能是因为在集成或维护阶段加入了新的模块。当软件中所含错误被发现时，如果错误跟踪与管理系统不够完善，就可能会遗漏对这些错误的修改；而开发者对错误理解的不够透彻，也可能导致所做的修改只修正了错误的外在表现，而没有修复错误本身，从而造成修改失败；修改还有可能产生副作用从而导致软件未被修改的部分产生新的问题，使本来工作正常的功能产生错误。

同样地，在有新代码加入软件的时候，除了新加入的代码中有可能含有错误外，新代码还有可能对原有的代码带来影响。因此，每当软件发生变化时，我们就必须重新测试现有的功能，以便确定修改是否达到了预期的目的，检查修改是否损害了原有的正常功能。同时，还需要补充新的测试用例来测试新的或被修改了的功能。为了验证修改的正确性及其影响就需要进行回归测试。

【回归测试的作用】

回归测试在软件生命周期中扮演着重要的角色，因忽视回归测试而造成严重后果的例子不计其数，导致阿里亚娜 5 型火箭发射失败的软件缺陷就是由于复用的代码没有经过充分的回归测试造成的。

回归测试作为软件生命周期的一个组成部分，在整个软件测试过程中占有很大的工作量比重，软件开发的各个阶段都会进行多次回归测试。在渐进和快速迭代开发中，新版本的连续发布使回归测试进行的更加频繁，而在极端编程方法中，更是要求每天都进行若干次回归测试。因此，通过选择正确的回归测试策略来改进回归测试的效率和有效性是非常有意义的。

2.8.2　人工测试与自动测试

人工测试与自动测试主要是从测试用例的执行者是人还是程序来区分的。自动测试由专用程序自动来测试目标软件，它具有如下特点：

- 节省时间，提高效率。可以定时定期批量运行，大大节省人力。
- 结果精确性好。对于同一测试用例，同样的程序测试同样的被测软件，结果一致

而且精确。设想验证网页上的播放视频窗口位置，假设要求其出现在浏览器窗口从左边框起 333 像素的地方，这样的话，人肉眼是无法精确验证的，而程序则很容易实现。

- 需要花费时间进行测试工具的开发或者学习。由于被测试的软件各有各的实际情况，测试工程师会根据自己的需求自定义开发专用程序来对软件进行测试；即便是市场上有通用的测试平台，也需要花费时间和金钱进行学习。这也就带来了一个风险，如果测试平台本身就不稳定，那么测试出来的结果也不会令人信服，有的时候甚至会误导测试工程师和开发人员，造成不好的后果。

因此，对于某些测试用例，如果利用现有的技术手段、资源无法快速而低成本地实现程序自动化测试，就只能由人工来完成，这样的测试就称为人工测试。

在实际工作中，自动测试必须和人工测试互为补充地进行，这样才能获得更好的测试质量和效率。

2.9　测试过程：有关项目和里程碑

对于初学者而言，软件测试可能不像头脑中最初想象的那样只在软件上市前进行，其实它处于项目开发的各个阶段。为了合理、高效地参与整个软件开发项目，测试工程师掌握一些项目管理相关知识也是很必要的。

2.9.1　测试贯穿整个项目流程

测试开始的越早，发现 Bug 并修正它所花的成本越少。确实，测试需要贯穿整个项目。在项目进展的具体各项步骤中，测试都需要做什么工作呢？

1．项目开始阶段

在项目开始阶段，整个项目组最重要的工作之一就是了解用户的需求，形成软件的说明书，并确定项目要采用的技术。在这个阶段内，测试工程师要做好如下的工作：

尽可能地熟悉用户的需求，这种了解不是类似项目经理一样亲自去客户那里，而是积极参与形成软件说明书的各种讨论；尽可能地熟悉项目开发的目的，为今后的测试工作打好基础。

2．项目设计阶段

在项目设计阶段，整个项目组最重要的工作之一就是确定项目的每一子模块的设计和实现方法。在这个时候，测试工程师所负责的子模块肯定也已经确定，要和负责该模块的软件开发工程师配合好，根据开发人员所采用的技术，对自己的测试有个计划。在可能的情况下，一起讨论并提高当前模块的可测试性。如果一个功能实现之后，测试起来非常困难，这对产品质量也会造成很大的风险。

在项目设计阶段结束的时候，软件开发人员应该有一份开发设计文档和开发计划，相应地，软件测试人员也应该有一份测试设计文档和测试计划。

3．项目执行阶段

这个阶段正是测试工程师大显身手的时期。

在项目实施阶段，整个项目组的工作重心都在编码完成软件功能之中。测试工程师此时要完成测试工具的编码与部署，保证软件测试环境的稳定，并定期发送最新代码的测试报告。

同时，在软件代码慢慢定型、项目进度慢慢接近尾声的时候，测试的重心也要随项目发展，从功能测试开始提升级别，到达集成测试、系统测试、性能测试、安全测试、兼容性测试等。

4．项目竣工阶段

在这个阶段，测试工程师要整理当前项目的测试技术和测试文档，整理测试经验，同时要关注维护阶段用户发现的新 Bug 等，协助项目组其他成员做好维护工作。

由这几个阶段可以看出，测试工作确实存在于项目自始至终的各个时期，而不是代码写完后集中一段时间的工作。

【各阶段还需要细分】

“罗马不是一天建成的”。实际的软件项目开发过程往往持续 1 年甚至更长。仅仅将项目分为上述的 4 个阶段依然是不好管理的（想想小时候临开学赶暑假作业的例子），最好能详细到每一天各工种完成什么事情才好。出于这样的目的，对项目各阶段进行细分就很有必要。

2.9.2　什么是里程碑

当我们开车行进在高速公路的时候，经常能够看到路边有牌子或者标记，显示目前的位置离终点或者下一个出口有多远，这就是里程碑，英文名叫做 Milestone。随着里程的变化，我们可以保持一直前进的动力，不至于迷失方向和信心。

【里程碑】

里程碑也同样存在于项目管理当中。在 2.9.1 节提到了简单划分项目的几个阶段，在其中，项目实施阶段一般来讲都是最长的，对于复杂的大型软件，甚至要跨越好几个年度。为了管理上的方便，同时提高效率，人们往往把大的阶段再划分为几个子阶段，这样实现起来更容易具体，有化整为零，各个击破的意思。这几个子阶段就形成了若干个里程碑，在工作中一般由 M1、M2 这样的字眼来表示：M 表示 Milestone 的缩写，后面紧跟的数字表示先后顺序。

在一个项目的进程中，里程碑的表示方法非常重要。虽然里程碑代表了一个子阶段，但在时间上它是一个点。比如，某项目设立了 3 个里程碑，如下所示。

- ❑ M1：在 2008 年 9 月 1 日前，网站内容部分编码完成。
- ❑ M2：在 2008 年 11 月 1 日前，网站用户部分编码完成。
- ❑ M3：在 2009 年春节前，网站开始集成与性能测试，完成优化，最终实现上线目标。

在这样的时间表中，各个日期就是各自子阶段的里程碑。

里程碑在整个项目中也不宜很多，否则反而容易使得项目参与人感觉项目漫长。在实际的工作中，一般都不超过 5 个，而且都是选择在项目产品发生明显变化的点上。

2.9.3　Project 软件中的里程碑

第 1 章介绍过项目管理软件 Project 中的甘特图，实际上，甘特图上也可以发现里程碑的身影，上面所说包含 3 个里程碑的网站项目在 Project 2003 中形成了一个甘特图，如图 2-7 所示。

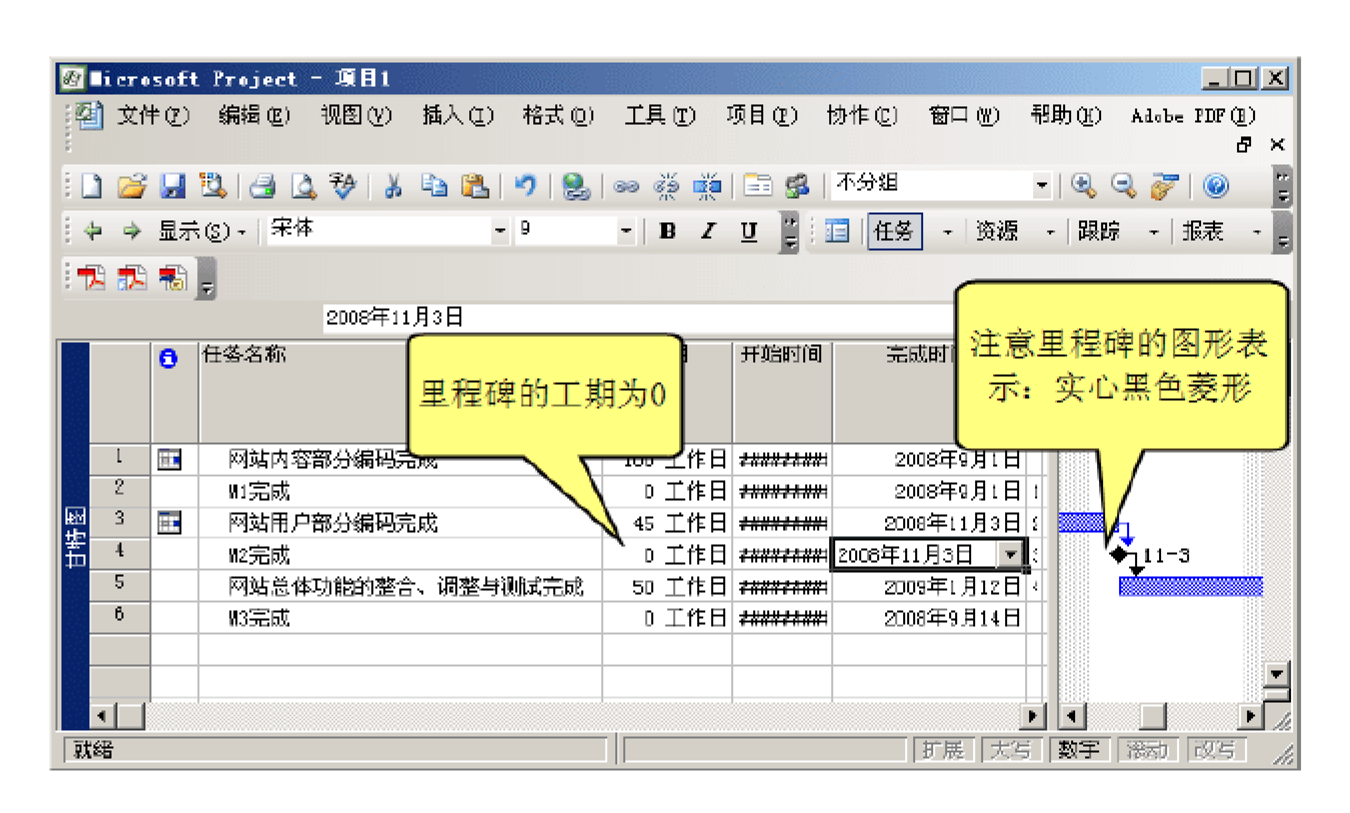


图 2-7　里程碑在 Project 2003 中的表示

在 Project 软件中，里程碑的插入一般是通过设置一个工期为 0 工作日的新任务来实现的。在甘特图中，它由一个黑色实心菱形来表示，并默认在旁边注明里程碑的日期。也有某些里程碑具备不为 0 的工期，但比较少见，请参考 Project 的相关书籍。

值得注意的是，每一个里程碑是否达到都有一个判定的标准，这个标准在划分项目的里程碑的时候就应该制定完成。

2.10　项目管理中的 ISO 9000 与 CMM

前面已经介绍了一些与测试工作相关的项目管理知识，下面再简要介绍一下在实际工作甚至在招聘广告中，经常能看到或者听到的两个英文缩写：ISO 9000 与 CMM。它们实际是两个标准的缩写。

2.10.1　ISO 9000 标准

最初的软件质量保证系统是在 20 世纪 70 年代由欧洲首先采用的，其后在美国和世界其他地区也迅速地发展起来。目前，欧洲联合会积极促进软件质量的制度化，提出了如下 ISO 9000 软件标准系列：ISO 9001、ISO 9000-3、ISO 9004-2、ISO 9004-4、ISO 9002。这一系列现已成为全球的软件质量标准。除了 ISO 9000 标准系列外，许多工业部门、国家和国际团体也颁布了特定环境中软件运行和维护的质量标准，如：IEEE 标准 729-1983、730-1984、Euro Norm EN45012 等。

由于软件开发与一般产品制造有显著的差别，因此必须注意软件过程的特点。可以想到，制造业或硬件的质量问题主要反映在生产和储运过程中，而软件的质量问题主要来自开发过程。所以，ISO 9000 国际标准在软件中的应用主要体现在以下内容：

- ISO 9001 质量体系是在软件设计、开发、生产、安装和维护时的质量保证的参考文件。此标准应用于所有软件产品和满足各种技术需求的软件维护活动中。它是评价软件质量的首要标准。
- ISO 9000-3 是对 ISO 90001 进行改造后，将其应用到软件工业中对软件开发、供应和维护活动的指导文件。
- ISO 9004-2 是指导软件维护和服务的质量系统标准。它指导和支持软件产品的维护。
- ISO 9004-4 是近年公布的很有用的附加标准，是用做改善软件质量的质量管理系统文件。

另外还有两个作为评价软件的标准：

- ISO 9002 适用于评价设计需求。此标准可以代替 ISO 9001，作为面向软件维护而不涉及设计的，为某些咨询公司、计算机培训及服务公司使用的基本标准。
- ISO 9003 适用于汇编及测试运行情况的标准。目前已经不再使用。

2.10.2　CMM 标准

美国软件工程研究所（SEI）开发的软件成熟度模型和国际标准化组织（ISO）开发的ISO9000 标准系列，都共同着眼于质量和过程管理，两者都为了解决同样的问题。

1．CMM产生背景

CMM 产生的背景主要是为了解决如下的问题：

在过去的 20 年里，新的软件开发方法和技术的使用并未使软件生产率和生产质量得到有效的提高。软件生产商开始意识到他们的基本问题在于对软件的生产过程管理不力，主要体现在：软件产品不能按时完成、超出预算的成本，以及采用新的技术和工具后其好处难以体现。

2．CMM的主要作用

CMM 可以指导软件机构如何控制软件产品的开发和维护过程，以及如何向成熟的软件工程体系演化，并形成一套良性循环的管理文化。具体说来，一个企业要想改进其生产过程，应该采取如下策略和步骤。

（1）确定软件企业当前所处的过程成熟级别。

（2）了解对改进软件生产质量和加强生产过程控制起关键作用的因素。

（3）将工作重点集中在有限几个关键目标上，有效达到改进机构软件生产过程的效果，进而可持续地改进其软件生产能力。

2.10.3　CMM 的一些基本概念

在 CMM 标准中，如下的一些概念会在不少文档中出现，因此有必要进行解释一下。

- 软件过程：人们在开发和维护软件及其相关产品时所涉及的各种活动、方法、实践和改革等。其中软件相关产品包括软件项目计划、设计文档、程序代码、测试用例和用户手册等。
- 软件过程能力：当遵循某个软件过程时所能达到的期望效果，它可以有效预测企业接收新的软件项目时可能得到的结果。
- 软件过程性能：当遵循某个软件过程时所达到的实际效果。它可以用于验证软件过程能力。
- 软件过程成熟度：指一个特定的软件过程被明确地定义、管理、测量、控制并且是有效的程度。成熟度可以用于指示企业加强其软件过程能力的潜力。 当一个企业达到了一定的软件过程成熟级别后，它将通过制定策略、建立标准和确立机构结构使它的软件过程制度化。而制度化又促使企业通过建立基础设施和公司文化来支持相关的方法、实践和过程。从而使之可以持续并维持一个良性循环。

2.10.4　CMM 的五级成熟度

制定 CMM 的五级成熟度，是从如下几点出发的：

- 软件质量在很大程度上取决于产生软件的软件过程的质量和能力；
- 软件过程是一个可管理、可度量并不断改进的过程；
- 软件过程的质量受到用以支撑它的技术和设施的影响；
- 企业在软件过程中所采用的技术层次应适应于软件过程的成熟度。

由于 CMM 系统强调连续的软件过程改进，而该连续的改进又基于多个演化步骤，为了更好的实施 CMM，CMM 将这些演化步骤划分成 5 个级别。每 1 级别都包括若干目标。当满足某 1 目标后，软件过程的相应部分便确定下来，有点类似于前文所叙述的里程碑。

五级成熟度定义了一个标准，用以度量机构的软件过程成熟度和评价其软件过程能力，目前主要涉及如下内容：

- 机构和资源的管理：涉及机构本身的责任，人员和其他资源设施。
- 软件工程过程及其管理：涉及软件工程过程，即软件过程的深度、范围和完整性以及如何度量、管理和改进这样的过程。
- 工具和技术：软件工程过程中使用的开发工具和技术。

【CMM 的五级成熟度具体级别】

CMM 的 5 个成熟度级别划分如表 2-3 所示。

表 2-3　CMM的五级成熟度

级　别	说　明	级　别	说　明
初始级	未加管理的过程	管理级	可预测过程
可重复级	有规章的过程	优化级	可持续改进的过程
定义级	标准化、一致的过程		

2.10.5　CMM 五级成熟度分级别详解

2.10.4 节的表 2-3 列出了 CMM 的 5 个成熟度，并且在“说明”一项中，以简明扼要

的文字阐述了各级别成熟度在过程管理上的特点。本节将展开表 2-4 中的说明部分，使读者对其具体意义有更深入的了解。

1．第一级：初始级

成功来源于个人英雄主义而非机构行为，因此它不可重复，更换人员后成功便难以维持。这是最低的一种成熟度。

2．第二级：可重复级

可重复级达到了如下的要求：

- ❑ 针对特定软件项目建立管理该项目的策略和实现这些策略的过程。
- ❑ 新项目的计划和管理基于类似项目的经验。
- ❑ 软件过程能力主要通过管理单个项目的软件生产过程来得到提高和增强。
- ❑ 不同的项目可有不同的软件过程，机构应当建立一定的方针和策略以针对具体的项目选择合适的软件生产过程并进行管理。

【可重复级的特点】

可重复级的主要特点在于确定了基本的软件生产管理和控制，具体来讲，就是结合已有项目的经验和新项目的特点来确定本项目的责任和承诺；软件生产成本、时间表和实现的功能被有效跟踪；识别实现承诺所需解决的关键问题；定义软件项目过程标准，机构要确保其被遵守这几项。

概括来说，第二级的主要特点是项目计划和跟踪是确定且有效的，项目的软件过程是可控的，以及已有的成功经验是可重复的。

3．第三级：定义级

定义级达到了如下的要求：

- ❑ 有一个机构范围内标准的软件过程，软件工程活动和管理活动被集成为一个有机的整体。标准化的目的是使高层管理者和软件技术人员能够有效合作。
- ❑ 有一个组例如软件工程组（SEPG）专门负责订立机构的标准软件过程，并且在机构中制定培训计划来确保相关人员和管理者有足够的知识和技能完成标准过程所赋予的角色。
- ❑ 标准的软件过程结合具体项目的特点经过裁剪即形成项目定义软件过程，它是一组集成的完善定义的软件工程和管理过程。
- ❑ 一个完善定义的软件过程应包括就绪准则、输入、工作过程、验证机制、输出和完成准则。
- ❑ 对于已建立的产品生产线，其成本、时间表和实现功能均可跟踪和控制，软件产品的质量可以得到保证。
- ❑ 软件过程能力的实现主要基于在机构范围内对一个定义软件过程的活动、角色和责任的共同理解。

【定义级的特点】

概括来说，第二级的主要特征在于软件过程已被提升成标准化过程，从而更加具有稳定性、重复性和可控性。

4．第四级：管理级

管理级达到了如下的要求：

- ❑ 软件的过程和产品有定量的质量指标。
- ❑ 重要的软件过程活动均配有生产率和质量方面的度量指标。
- ❑ 应用数据库来收集和分析定义软件过程中涉及的各种数据。
- ❑ 对项目软件过程和软件质量的评价有定量的基准。
- ❑ 软件项目的产品和生产过程的控制具有可预测性。
- ❑ 将软件过程性能可能出现的偏差控制在可接受的量化界限内。
- ❑ 具体区分影响过程性能发生偏差的有效因素和偶然因素。
- ❑ 向新领域拓展的风险是可预知的并被仔细管理和权衡。

【管理级的特点】

概括来说，第四级的主要特征是定量化、可预测、异常控制和高质量。

5．第五级：优化级

优化级是最高的一个级别，它达到了如下的要求：

- ❑ 机构集中于持续的过程改进。
- ❑ 具有标识过程缺陷和增强过程能力的有效手段。
- ❑ 利用试验数据分析使用新技术所需的代价和带来的效益，然后再有选择地采用。
- ❑ 当出现偏差时，软件项目人员能够分析出错原因并采取有效手段防止其再次出现。
- ❑ 防止不必要的浪费是第五级的重点。改进的途径有两个，一个是对已有过程的渐进式改进；另一个则是有选择地使用新技术和新方法所带来的革新。

【优化级的特点】

概括来说，第五级的主要特征是新技术的采用和软件过程的改进被作为日常的业务活动来加以计划和管理。

值得一提的是，软件成熟度已经被 ISO 组织接纳，成为 ISO 15504 的一部分。

2.11　软件测试中的维恩图

下面再补充几个测试工作中经常用到、却不引人注意的知识点。在本节中简要介绍一下维恩图在软件测试中的使用。在实际工作中，特别是测试讨论会议的场合，用维恩图这一简单的图形可以使得大家对于当前的测试状态一目了然。

2.11.1　维恩图简介

看到这个题目可能读者会感觉比较高深，其实维恩图和前文所讲的甘特图类似，都是以人名来命名的。

还记得初中的时候学习集合吗？那时候交集、并集等名词对当时的我们来说不很好理解，这时候老师就会在黑板上画一个图，用两个圆来表示两个集合，如图 2-8 所示。

通过这样的图一看，有关交集并集的概念就很容易理解了。

【软件测试中的维恩图】

利用传统的维恩图，我们可以比较直观地了解到程序行为、软件说明书和测试用例之间的关系，从而大致确定当前测试的覆盖率，对被测试的软件质量有个大概估计，为下一步发现 Bug，增加测试用例找到一个突破口。请看图 2-9，它表示了当前测试的进度。

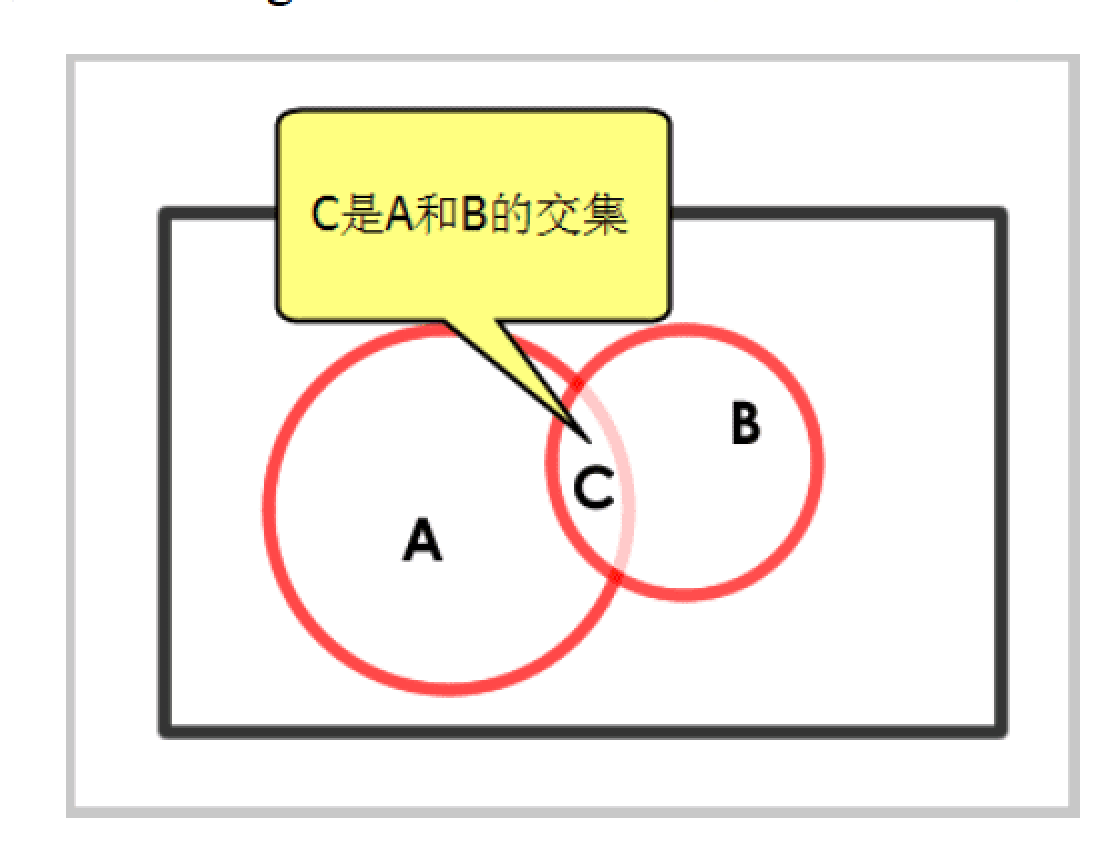


图 2-8 维恩图示例

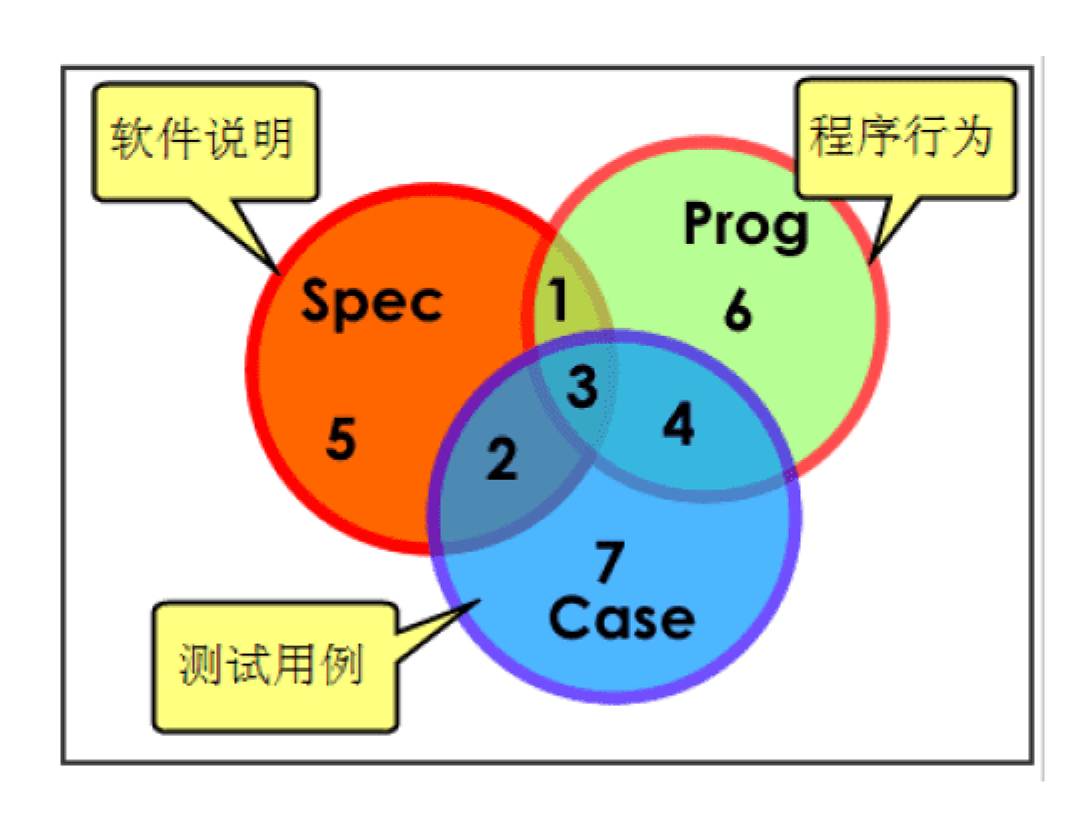


图 2-9 描述软件说明、程序行为和测试用例关系的维恩图

2.11.2 软件测试中的维恩图详解

下面举例说明维恩图在软件测试中的使用方法。

在图 2-9 中，分别由 3 个圆代表软件说明的所有内容、程序的所有行为和所有的测试用例。由于 3 个圆的相互部分重叠，导致维恩图中出现了 7 个部分，分别标示在图中。这 7 个部分分别代表如下的含义。

- ❑ 数字 1 代表的部分：没有测试用例覆盖的在软件说明书中存在的程序行为。
- ❑ 数字 2 代表的部分：有测试用例覆盖，在软件说明书中也提到，但程序行为不可能出现。
- ❑ 数字 3 代表的部分：代表测试用例已经覆盖，在软件说明书也提到并且实际程序也有这样的行为。实际上就是当前测试过的那部分软件功能。
- ❑ 数字 4 代表的部分：已经测试过的，但在软件说明书中没有提到的那部分软件行为。
- ❑ 数字 5 代表的部分：在软件说明中出现，但尚未测试过，而且程序不可能出现的行为。
- ❑ 数字 6 代表的部分：未被测试，而且软件说明书中也未明确的程序行为。
- ❑ 数字 7 代表的部分：这部分的测试用例既没有覆盖到程序行为，也没有依据软件说明书，基本属于无效测试用例。

从以上各部分可以很快明白，作为一名测试工程师，应该：

- ❑ 尽量减少无效的测试用例，以保证工作的有效性，即数字 7 的部分要尽量小。
- ❑ 区分不同优先级的测试用例，比如对于数字 5、数字 2 代表部分的测试，如果软件说明无法更改，可以将其优先级放置较低。

- 尽量提高已经测试过的数字 2 部分的面积，这标志着当前软件已经完成的测试覆盖率。
- 明白下一步测试的目标，重点就是数字 1 所代表的部分直到消失。

可见，利用维恩图，我们对当前的测试工作就会有一个总体的把握，因此在各种各样的工作讨论场合中很有用处。不论是具体的单元测试或者功能测试，还是整体的项目进展，维恩图都可以比较直观地总结现阶段没有完成的工作，并提示哪些地方是下一步的重点。

2.12　两组容易混淆的测试用语

前面已介绍了国际化与本地化这一对容易混淆的测试术语，本节再区分两对测试用语，分别是精确性与准确性；验证合格与确认可用。它们并不是严格意义上的术语，但是经常被测试工程师挂在嘴边，因此明确理解其含义是很重要的。

2.12.1　精确性与准确性

精确性和准确性是测试工程师实际工作中经常要使用的两个词语之一。设想下面这样一个例子：

小白要测试的网页上有一个计算器，用来计算每月工资应缴纳个人所得税的情况，浏览者输入数字进行计算，网页返回结果，如图 2-10 所示。那么，当测试的时候，如何说明它的结果是精确还是准确呢？

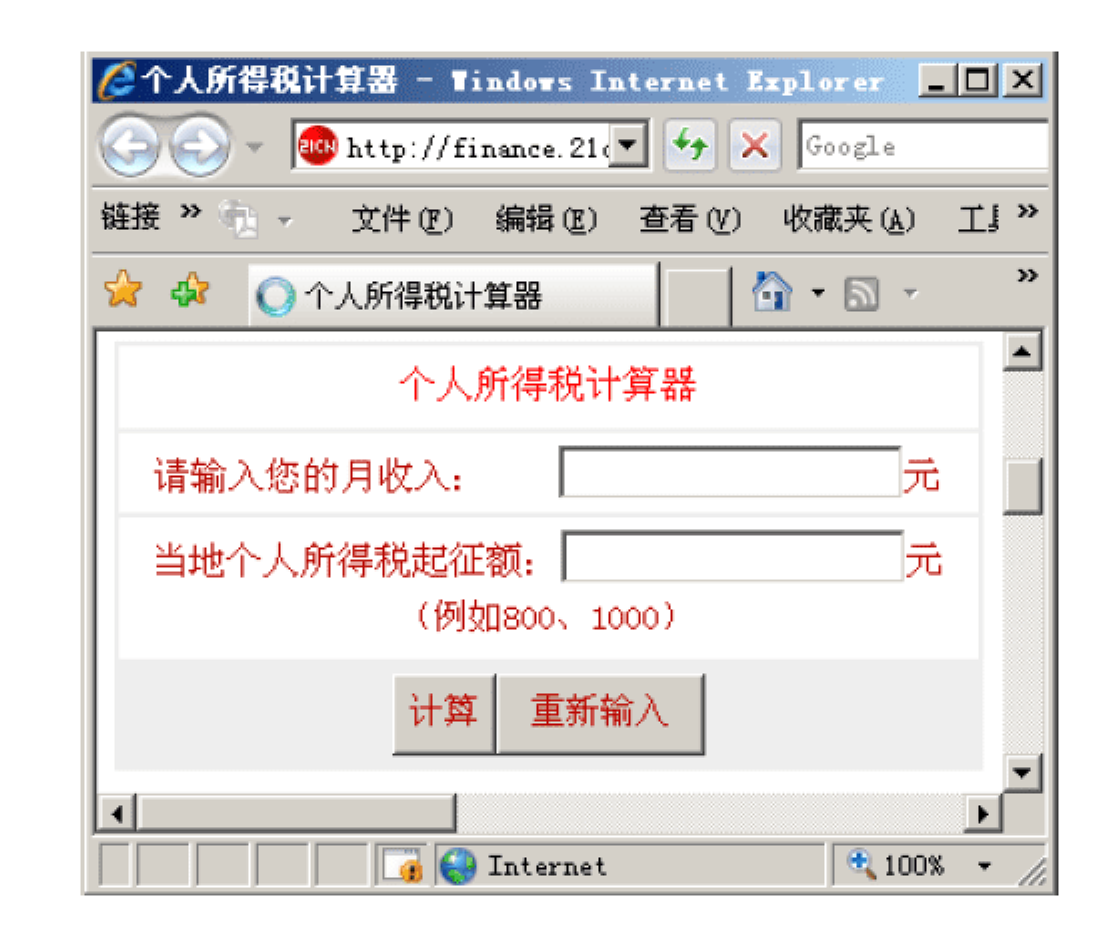


图 2-10　一个网页上的个人所得税计算器示例

实际上，这两个词在生活中的用法是基本混淆的，但如果在软件测试中使用，需要明了二者具体含义的区别。下面以本届奥运会中国队员获得突破的射箭项目来解释这个区别。如表 2-4 是某场射箭比赛 4 位运动员分别射出 3 箭的成绩单。

表 2-4　射箭比赛成绩单

选　　手	第 1 箭成绩	第 2 箭成绩	第 3 箭成绩	评　　价
选手 A	脱靶	4 环	2 环	不精确，也不准确
选手 B	1 环	1 环	1 环	精确，但不准确
选手 C	8 环	9 环	7 环	准确，但不精确
选手 D	10 环	10 环	10 环	精确而又准确

由此可见，精确描述多箭成绩之间的差异：在选手 B 的成绩单中，由于 3 箭成绩一样，所以成为精确。准确描述多箭成绩与目标之间的差异，选手 C 每箭和靶心的差距都比较接近，因此说具有一定的准确性；而选手 D 每箭都是 10 环，既精确又准确。

将这个例子扩展到前文所说的网页上税计算器，如果计算结果每次都一样，那么可以说是很精确；如果计算结果和实际上税数量一样或者在允许的误差之内，则可以说是很准

确。当然，两者都满足是这个网页应该达到的目标。

在实际测试过程中，经常会发现结果很精确，但不准确的情况。这其中绝大部分的原因都是由于软件 Bug 引起的，需要我们特别注意。因此，掌握好精确和准确两个词的含义，还是很有必要的。

2.12.2　验证合格与确认可用

验证合格与确认可用的英文分别对应于 Verification 和 Validation，由于中文很难找到专门的两个术语以显示区别，因此笔者采用了“验证合格”和“确认可用”这样的说法。这两个词语在实际的工作中，特别是外资企业应用较多。而且，沟通中直接使用它们的英文单词会更方便和易于理解。

虽然这两个词语英文拼写和中文翻译都很类似，但是它们具体含义上的区别还是很明显的，当然也很重要。

所谓验证合格，即 Verification，指的是软件行为符合说明书（经常提到的 Spec）的规定。而确认可用（Validation），指的是软件行为满足用户的需求。这两者在理想状态下是应该完全一致的，但不要忘记，软件的说明书也是人来写的，用户的需求也是人来发现的，因此，二者会有偏差，很可能出现软件行为符合规范，却不满足用户需求的情况。

那么，最终以什么为标准呢？答案很简单。用户是上帝，自然是以满足用户的需求为标准。在很多公司的测试工作中，确认合格的过程更多的是以自动化测试来完成；而确认可用的过程，则相对较多地以人工测试或者集成测试来完成。作为 1 名测试工程师，必须知道二者的区别，并且在工作中将它们有机地结合起来。

2.13　本 章 小 结

本章讲述了有关测试的一些基础知识，跨度较广。在本章的开头，主要介绍黑盒测试和白盒测试的概念，重点介绍其中设计黑盒测试用例的方法，主要有：

- 等价类划分；
- 边界值分析；
- 决策表。

黑盒测试主要用于功能性测试，对测试工程师的编程知识要求并不高，完全以外部的角度观察软件行为，更贴近用户使用场景，在实际工作中应用很广。除了黑盒测试之外，白盒测试也非常重要，它广泛地应用于开发工程师的单元测试、代码覆盖度测试中等。

本章还简要介绍了性能测试与压力测试，本地化与全球化测试，回归测试等实际工作中经常碰到的术语。在本章的后半部分，对于测试工作中经常遇到的里程碑、维恩图使用、项目管理与 ISO 9000、CMM 的知识也有所涉及，这些都是一个合格的测试工程师所需要了解的知识。

第 3 章将介绍针对 Web 网站的性能测试。

第 3 章　Web 应用与 Web 测试

在本书的前两章，笔者已经对测试的两个重要组成部分：软件测试工程师、测试用例及其编写的方法进行了简要介绍。从第 3 章开始，本书将进入 Web 测试的世界。首先，3.1 节将讲解 Web 应用的一些知识，进而介绍有关 Web 测试的诸多分类，比如功能测试、性能测试、兼容性测试等。

3.1　Web 应用的基本知识

“万丈高楼平地起”。在展开 Web 应用的性能测试之前，首先需要了解有关 Web 应用的若干基本概念，并熟悉建立网站，以及浏览网站的基本过程，这些知识和经验对于理解和解决实际发生的 Web 应用性能问题会有较大的帮助。

本节通过“请客吃饭”这一场景类比用户浏览网站的过程，使读者能够熟悉服务和 Web 服务的概念。

3.1.1　什么是服务

在 Web 应用中的网页文件与 Word 文档等其他类型的文件一样，都是存放于电脑的硬盘之中。但是，与供用户自己使用的文档不同，绝大部分的网页都是供非编写者浏览的。为了让使用者能够通过网络能到 Web 应用开发人员辛辛苦苦编写出来的网页文件，首先须把存放网页文件的电脑变成一台能够提供网页浏览服务的服务器。

通俗地讲，IT 世界里的“服务”与日常生活中所遇到的“服务”这两个词语在含义上没有什么本质区别。假设读者要请自己的朋友吃饭，那么一般会有两种选择：在家里亲自下厨招待或者去外面的饭店。下面笔者不妨详细地比较一下这两个选择，从中引申出 IT 术语“服务”的概念和相关含义。

在自己家每天做饭吃，可能不需要太过讲究，只需考虑营养问题、做饭是否快捷、饭菜是否合乎自己的胃口等几个要点就行了。但是，如果请朋友到家里吃饭，主人就必须得有一点服务的观念，除了之前考虑的那几个要素，更要充分考虑朋友的喜好和要求，一句话，请朋友吃饭，是为了朋友能够吃好，这样才能增进彼此的感情。

开发 Web 应用也是同样的道理：技术人员制作出来的网页是要交由使用者进行操作、浏览者查看信息，因此要尽可能地满足最终用户方方面面的需求，可以说，IT 行业中所说的服务，与请朋友吃饭非常类似，具备同样的出发点。满足用户需求就是“服务”这个词语的主要含义。

【服务的 IT 定义】

服务，是一个或多个程序的集合，为别的程序或最终用户提供所需要的信息。

3.1.2 服务的场所

无论日常生活还是在电脑世界里，服务总是有个请求→反馈的过程，这个过程构成了服务这个词的具体内容。

“民以食为天”，前文提到过请朋友到家里吃饭的选择，在当今的时代，更多的是请朋友到外面的饭馆去吃，这是因为饭馆的服务更专业：由专业的厨师进行烹饪，由专业的服务员端茶、送水、上菜，更有比家庭更适合用餐的专业环境等。同样地，在 Web 应用领域，为了满足用户的需求，也要专业、到位的服务。类似“饭馆”这样可以提供专业服务的场所，在 IT 领域是由硬件与软件相结合共同实现的，提供服务的硬件设备就是“服务器”（Server），提供服务的软件并没有专门名称，就称为“服务”（Service）。

在本节，将主要介绍提供服务的硬件场所——服务器。如图 3-1 是某品牌服务器的外观。

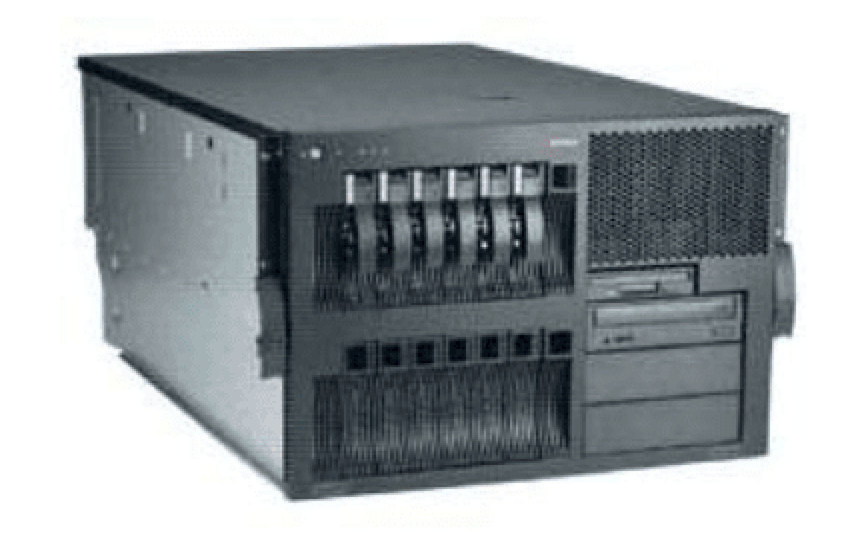


图 3-1 某型号服务器的外观

可以看到，服务器与普通台式电脑差别不大，其在硬件上同样包含各种各样的接口、插槽、连线等。但是，为了满足多种用户的不同需求，服务器对普通的台式机器、笔记本电脑进行了优化与改进，具备如下特点。

1．更大更强的能力

这个特点主要是指服务器具有速度更快的 CPU，容量更大的硬盘，速度更快、容量更大、缓存更多的内存等。为了解释这一点，笔者再次举出“请客吃饭”的例子。

一般来说，饭馆都要比家里地方宽敞很多：有不少餐桌，这样才能容纳更多的顾客来就餐。有时候，当用餐者来到某家很火的饭馆，如果位子不够，也就是饭馆容量不够大，服务人员会提示需要等待，一旦时间长了用餐者就会产生抱怨的情绪，从而影响服务的效果。同理，服务器如果硬件条件不好，对于网民的浏览页面请求无法完全满足，一样会导致服务效果的下降：网站打开很慢、出现内部错误等。所以说服务器的硬件设施一定要好，才能够提供更好的服务。

2．服务器中运行着服务

这里的服务是指提供服务的软件，即 Service。

饭馆里只有大间的屋子，众多的餐桌，如果服务员态度恶劣，那么这个饭馆也坚持不了多久。对于服务器来说，也是同样的道理。除了良好的硬件条件以外，服务器上还必须运行一些服务程序。我们先来看一下服务员是如何工作的：他们要随时查看每个桌子顾客的需要，然后采取相应的措施：上茶、上菜还是结账等。更好的状况是，他们中间有一部分人随时查看饭馆门口的情况，如果客人满了，要给没有座位的顾客以安抚，好的饭馆还会发给一些号码，以提示用户等待的时间等。

好的服务程序也是一样，能够按请求服务的程序或者用户的需要反馈回相应的合适信

息。它相对于坏的服务程序还有一个优点：能够在有限的容量内尽量实现更多的请求。（想想同样是 10 张桌子的饭馆，一个手脚麻利的服务员和一个呆若木鸡的服务员工作效果的不同）。当然，如果顾客盈门，实在太多，好的服务是应该给“没有座位的顾客”排队号码，从而在有空余的时候尽早对他们进行服务，以避免因为不满的顾客太多使饭馆的声誉下降甚至出现崩溃等严重的危机。

本节用饭馆类比服务器，服务员类比服务程序，充分说明计算机世界和日常生活是很相像的，不少道理都是相通的。今后在实际工作中遇到难以理解的问题时，不妨联系日常生活进行一番思考，可能会加深对问题的理解。

3.1.3 节起将开始介绍为 Web 应用创建服务场所的具体操作：在服务器上，使开发完毕的软件代码能够为满足用户需求而服务的过程。这个过程简单地说，就是网站的建立。

3.1.3　创建服务场所——建立网站

建立网站就是一个创建服务场所的过程。

继续以“请客吃饭”为例。人们都知道，开饭馆需要考虑很多种要素：要考虑市场（价位如何）、面对的群体（商务宴请还是小区居民）、菜系（四川还是广东或其他）、场所（租金地段是个大问题）、厨师等人员（工资福利）。网站的建立也是一样的辛苦：要考虑网站的规模定位（可以是大而全的门户网站，也可以是小而精的专业信息网站等）、面对的群体（个人网络购物，个人交友社区还是作为展示单位资讯的窗口，让潜在客户获得有价值的信息）、“菜系”（实现网站所采用的技术是什么？），场所（网页存放的服务器性能如何，用户流量大不大，方便不方便用户访问等诸多问题）、技术人员的分工等。当然，对于普通的性能测试工程师来说，不必考虑那么多的方方面面，但心中必须明确的观念是：网站是众多人员合作的产物。

在技术方面，测试工程师则需要知道一个网站是如何被建立的。创建网站，笼统地划分可以有如下 3 个步骤。

（1）建立网页的过程。技术人员需要利用多种编辑工具（记事本、写字板、VI 等程序）或者专业的网站开发工具（Dreamweaver、Visual Studio 等）将设计师设计的网页效果图变为现实。同时，网站还需要数据库等多方面的支持才能更好地满足用户需求。

（2）规划服务器空间的过程。这个过程视网站规模大小而有所差别。如果是个人主页或者是博客，则可以申请网络服务提供商（Internet Service Provider，简称 ISP）提供免费的个人主页空间，以及利用某些网站提供的模板自定义；如果是小型网站，可能要到 ISP 购买相应的服务器空间；对于大中型网站，一般则是把自己的服务器直接放到网络服务提供商的机房（又称为“互联网数据中心”，IDC）中，利用 IDC 所提供的网络出口对外提供服务。

【提高服务质量的方法】

对于大中型网站，往往有多台服务器可以提供带冗余的服务（一台服务器出现问题，马上由另一台服务器接替工作）以保证服务质量，这个方法有点类似增加候补队员以提高“板凳深度”。另外，由于客观现实的要求，会有不少网站在各大 ISP 和网络（比如教育网、电信网、甚至局域网）中都会放置服务器，存储相同页面，提供相同服务，因此可以保证处于各个网络内的用户都能正常顺畅地浏览和使用。这是另一种提高服务质量的方法，有

点类似大饭馆在全国各地开的连锁店面，使得各地的食客不必远行。

（3）发布网页的过程。有了存放网页文件的服务器空间，还需要一个发布网页的过程，网站才能最终建立。简要地说，要通过各种方式，把本地（指开发 Web 应用代码的电脑）编辑好的网页代码传输到服务器的硬盘空间。之后，还需要开启在服务器上运行的网页发布服务，从而完成发布网页的过程。这一过程将在 3.1.4 节中介绍。

3.1.4　网站文件的上传

编辑好的网页代码从本地电脑转移到服务器空间的过程就是上传（Upload）。上传可以通过多种方式来进行，比如文件传输协议（FTP），共享文件夹复制与粘贴等。其中最常用的则是 FTP 方式。

FTP，全称是 File Transfer Protocol，文件传输协议。我们将在下一部分讲述协议的故事，这里读者只需要知道它是一个可以用来上传文件的国际标准就可以了。而且我们不需要了解它的具体内容，因为有很多支持这个标准的软件可以供我们使用。比如 LeapFTP、CuteFTP 等。利用这些软件的功能，我们就可以实现网页的上传。

下面我们以 CuteFTP 为例来说明如何从本地电脑将网页文件上传到服务器的过程。

（1）运行 CuteFTP，按下 Ctrl+N 键，按照图 3-2 所示，设置申请服务器空间时 ISP 提供的 FTP 上传地址、用户名和密码。

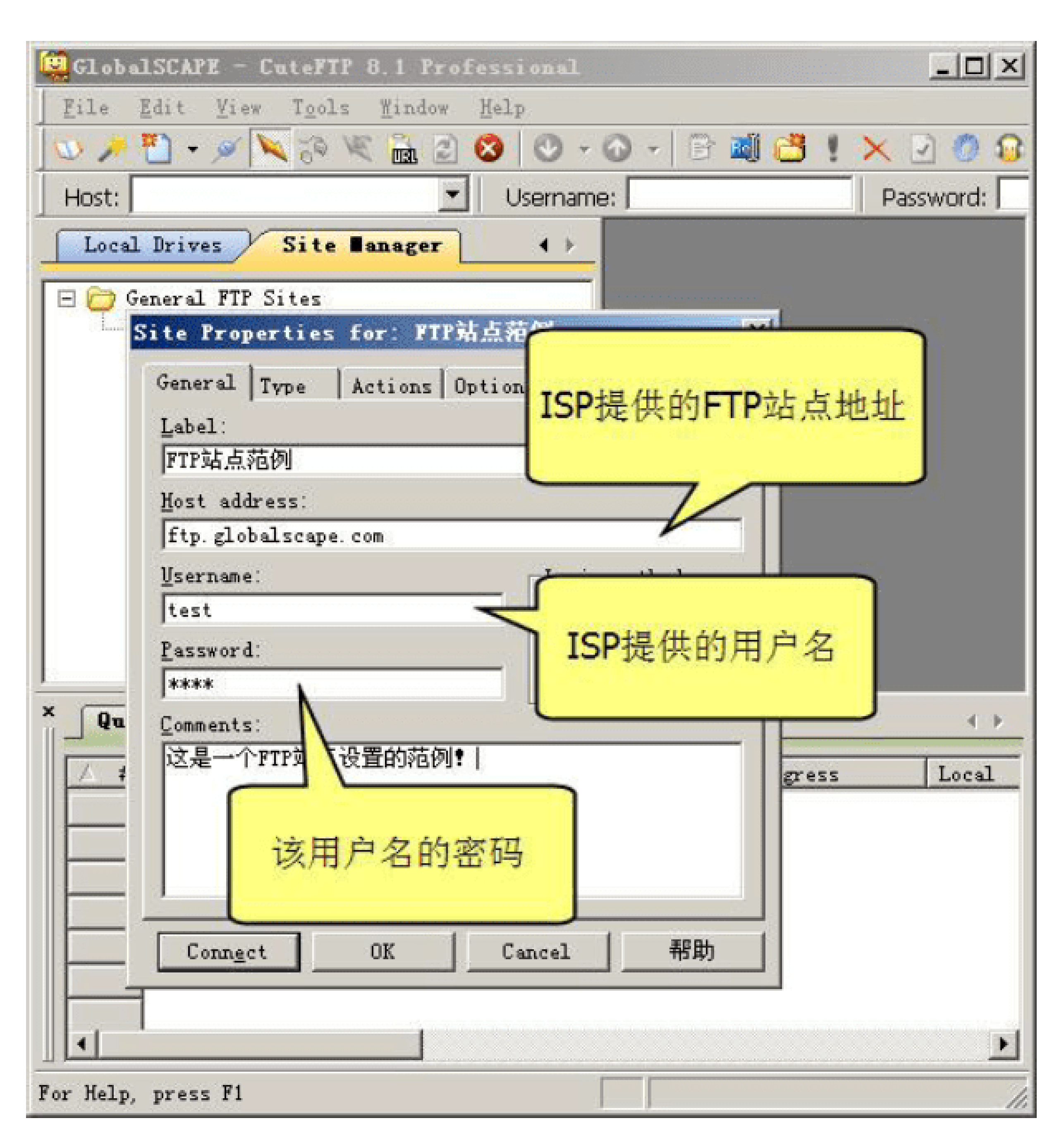


图 3-2　在 CuteFTP 8.1 中设置一个新的 FTP 站点信息

（2）单击 OK 按钮后，可以看到 CuteFTP 主界面分为两个部分，如图 3-3 所示。这两个部分分别代表本地计算机和服务器上的空间。具体上传步骤很简单：在“站点管理”（Site Manager）选项卡内双击新建立的 FTP 站点，连接成功后将本地硬盘上已经完成的网页文件拖动到服务器空间的相应文件夹中就可以了。

其他 FTP 软件基本上都是类似的界面和操作。

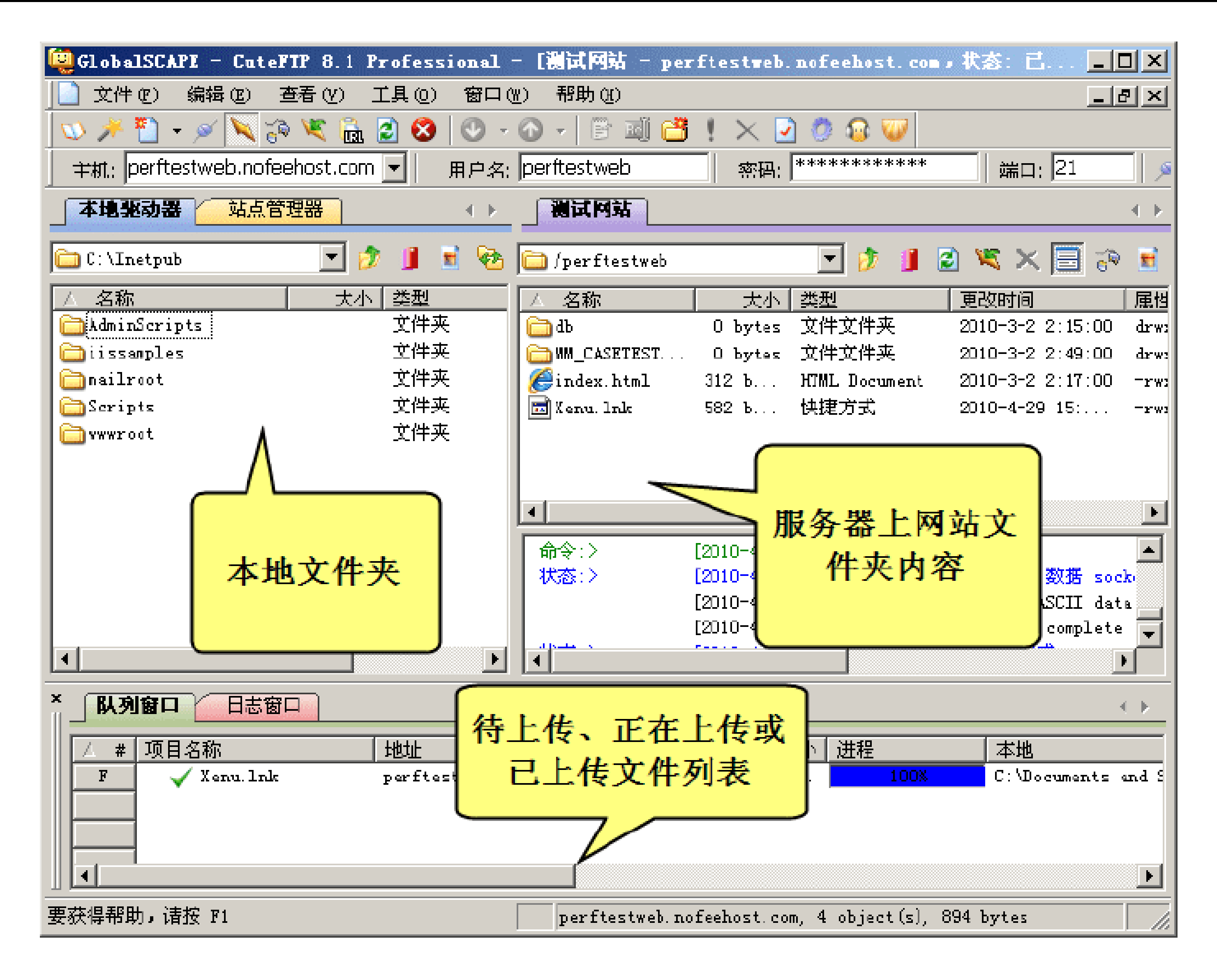


图 3-3　CuteFTP 界面

还有一些网页开发/编辑软件集成了网页的上传功能。下面以业内使用比较普遍的 Dreamweaver 和 Visual Studio 2005 为例说明网站集成开发环境（也叫做 IDE）中的上传操作。

Dreamweaver 中首先要建立一个站点，按图 3-4 和图 3-5 所示分别设置好本地与远程服务器空间的信息，比如服务器空间的用户名和密码，然后就可以进行编辑网页的过程。编辑结束后，可以直接选中文件并右击，在弹出的快捷菜单中选择 Put（上传）命令即可。

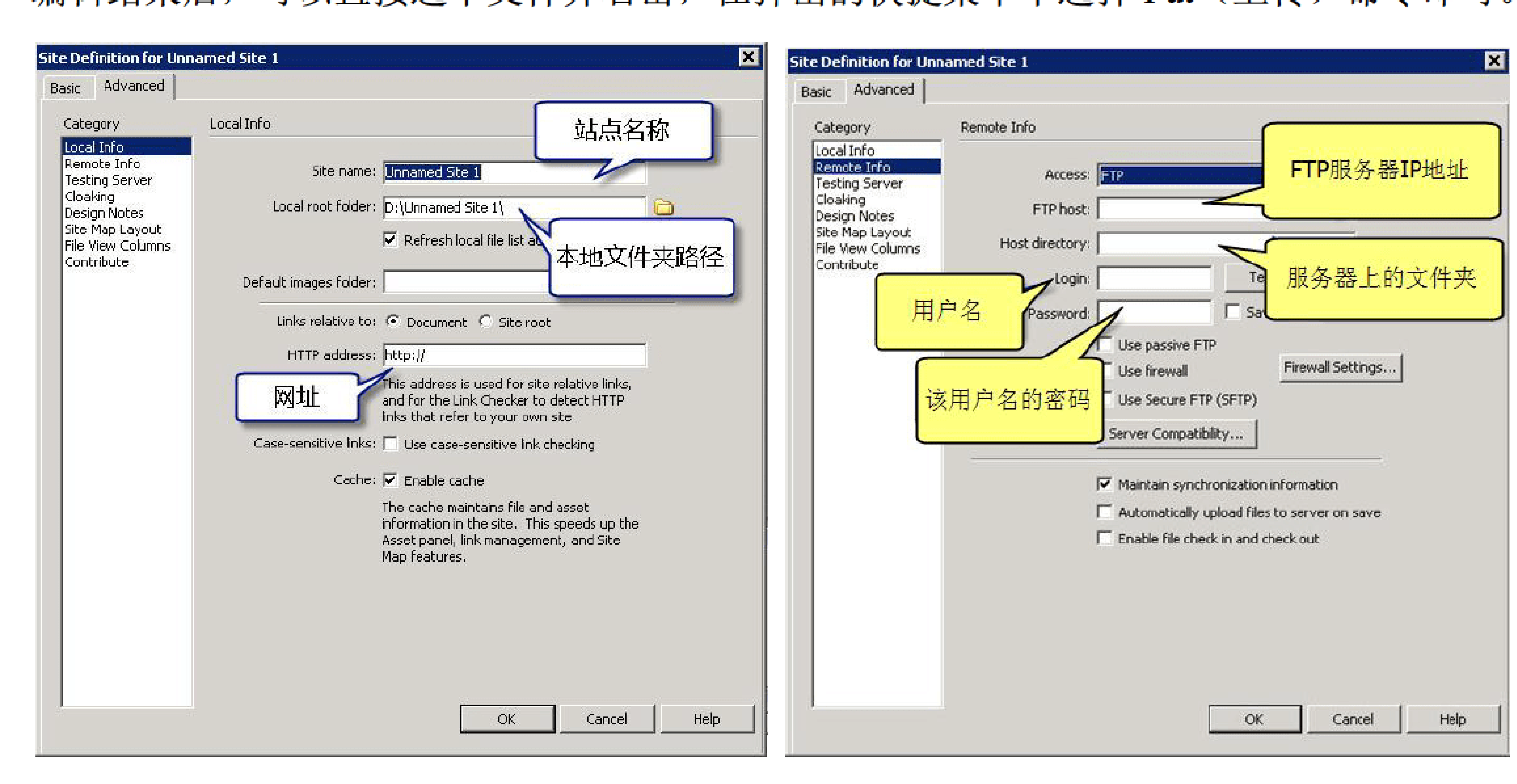


图 3-4　Dreamweaver 8 中的站点本地设置　　图 3-5　Dreamweaver 8 中的站点服务器信息设置

要完成同样的上传工作，在 Visual Studio 2005 中的操作有所不同，本书以其 Team Suite 版本为例。首先需要新建一个网站（依次选择“文件”|“新建”|“网站”命令），在完成网页的编写后，选择 Build|Publish Web Site（生成|发布网站）命令，Visual Studio 2005 会弹出如图 3-6 所示的对话框，在 Target Location（目标地址）文本框中填写正确的域名并

单击“确定”按钮后，若当前登录账户对于服务器文件夹有写入的权限，网页就随之上传到了服务器硬盘的相应目录中。

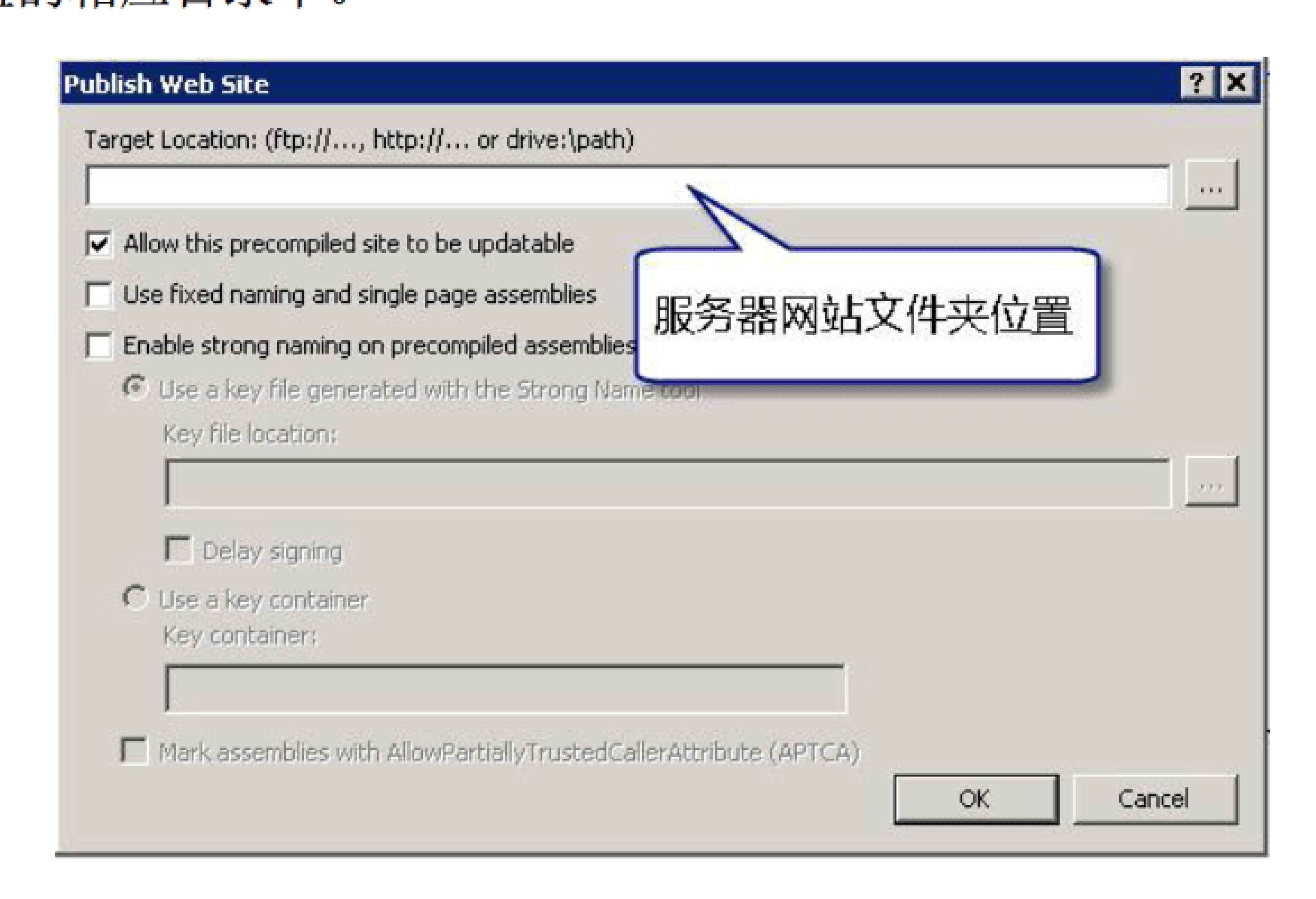


图 3-6　在 Visual Studio Team Suite 中发布网站

3.1.5　开启网页发布服务

在服务器上有了网页文件就相当于饭馆的厨师做好了饭菜，还需要服务员将其端到食客的餐桌上。这个“服务员”的功能是由运行在服务器上的网页发布服务程序来完成的。由于历史和商业的原因，这样的程序有很多，比如 Windows 平台下的 Internet Information Services（简称 IIS），以及支持多种操作系统的 Apache 等。如图 3-7 显示了在 Windows 服务器中打开 IIS 管理器后的界面。当然，IIS 管理器也可以通过单击“开始”按钮，在弹出的菜单中选择“运行”命令，并在文本框中输入 inetmgr 来启动。

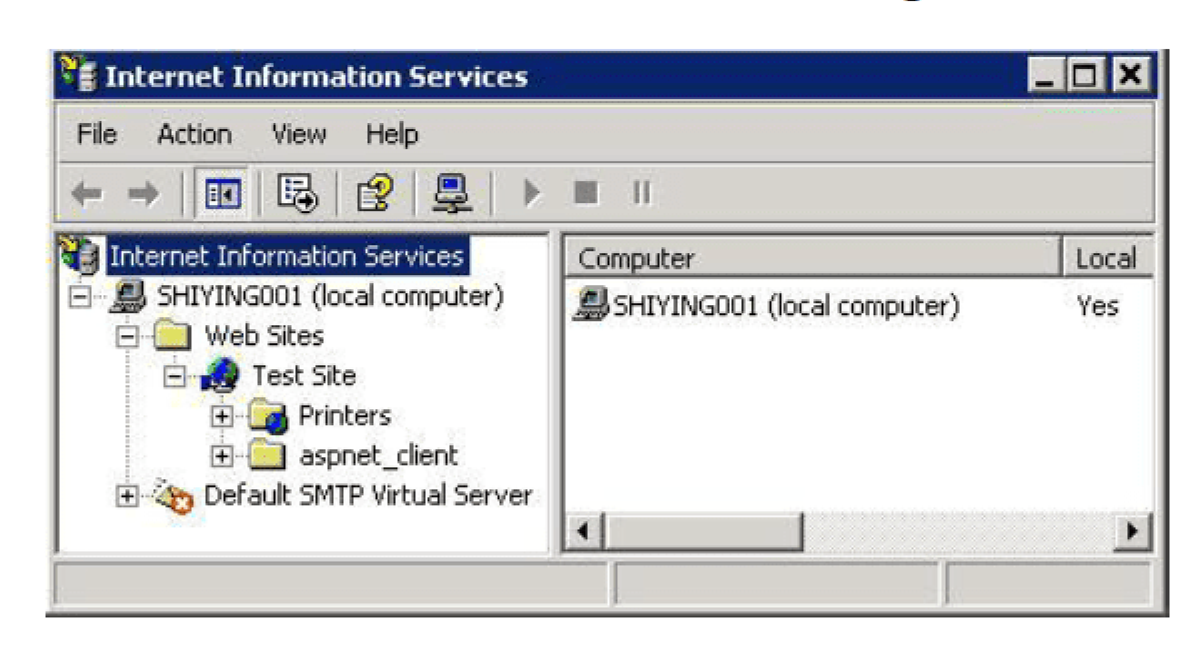


图 3-7　IIS 管理器界面

（1）如图 3-7 所示，在 IIS 管理器中会包含服务器的名字，其后面的括号中还有 Local Computer（本地计算机）的字样。单击该服务器名称，右击 Web Sites（网站）节点，在弹出的快捷菜单中选择“新建”|“站点”命令，建立新站点向导的界面就会出现。单击 Next 按钮后，界面如图 3-8 所示。

（2）在图 3-8 中的 Description（描述）文本框中输入待建立站点的说明，比如：“XXXX 公司网站”这样的文字即可。从这一设置页开始，剩下的几个页面均可保留系统提供的默认值，不做任何修改，连续单击多个 Next 按钮，直到图 3-9 所示界面的出现，整个创建过

程基本完成。



图 3-8　建立新网站向导：输入新建网站的描述信息

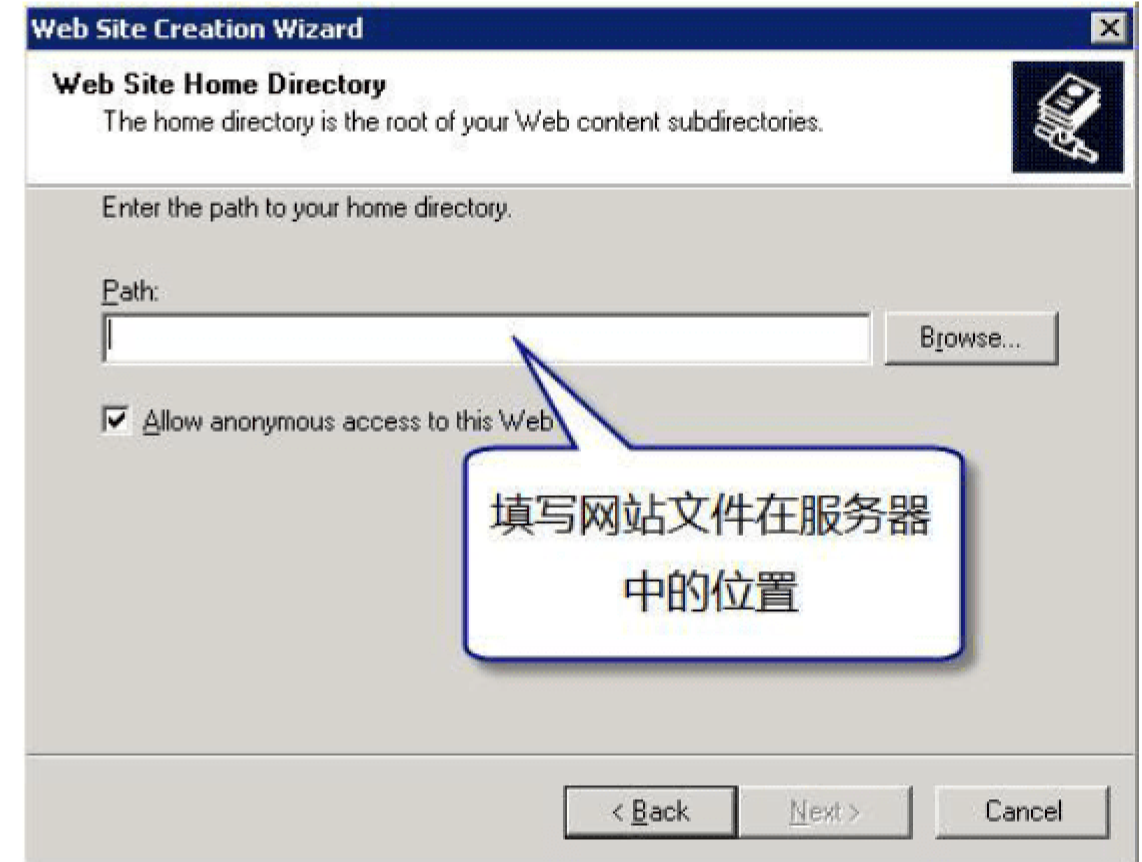


图 3-9　建立新网站向导：输入网站文件在服务器中的位置

（3）根据实际情况在图 3-9 所示界面上设置好网站文件所在服务器的具体位置（单击 Path（路径）文本框右边的 Browse（浏览）按钮，选择网页文件的根目录即可），继续单击多个 Next 按钮，直到最后向导提示完成。这样，一个网站就建设好了。

（4）除此之外，还要确认新建的网站是否启动了发布：选择新建立的网站，观察 IIS 管理器工具栏中的“运行”按钮是否为虚，如图 3-10 所示。

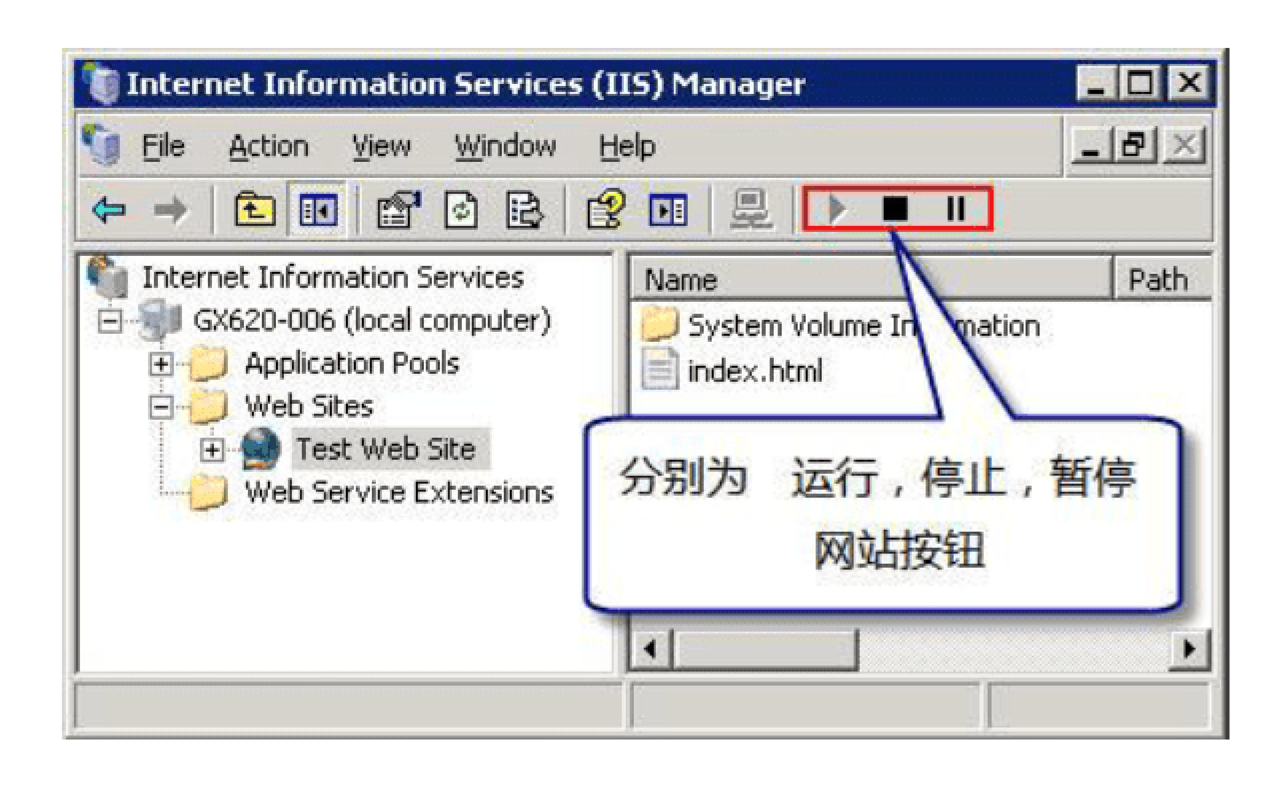


图 3-10　在 IIS 管理器中察看网站是否运行

建立好了网站，也开启了服务，那么如何确定我们编辑好的网页显示符合我们的期望呢？方法也很简单，在 IIS 管理器左边的树形目录中选择刚刚建立好的网站右击，在弹出的快捷菜单中选择 Browse（浏览）命令，就可以在右边的内容区域中看到劳动成果了。

除了 IIS 之外，前文提到还有很多的网页发布服务可供使用，比如 Apache 等，其需要设置的网站属性与 IIS 区别不大，但必须手工修改配置文件。因此，在实际操作方面，IIS 提供了更丰富细致的管理界面，初学者更易直观地理解。

3.1.6　用户浏览网站的过程

运行在服务器端的网站截至目前已经架设完毕，也就是说，饭菜已经端上了餐桌，剩

下的就是品尝了。本节将介绍在对“网站”的品尝当中，用户端所经历的过程。如图 3-11 是一幅示意图，显示了用户浏览网站的粗略过程。

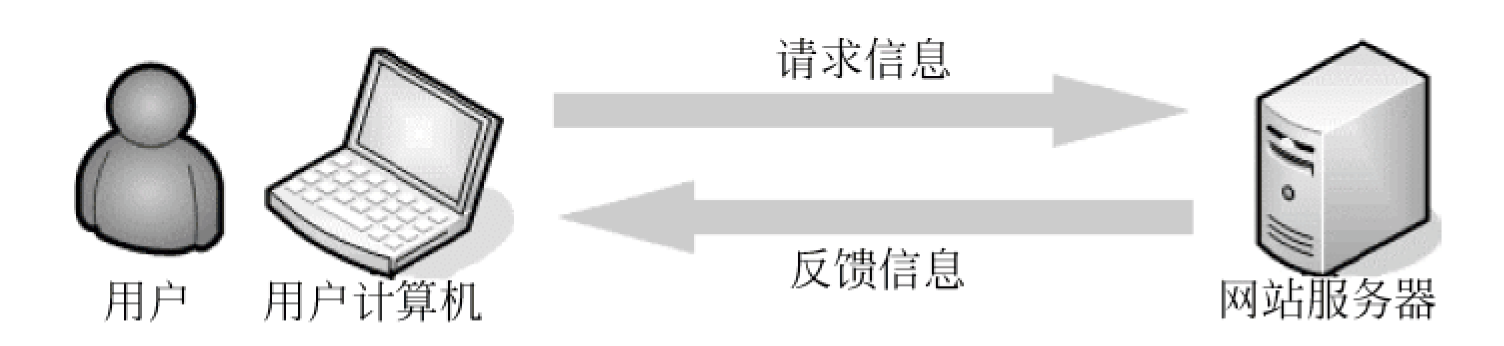


图 3-11　用户浏览网站的基本过程

下面把整个过程分成几个步骤来详细地解释。

（1）用户的操作：用户打开本地的浏览器，在地址栏中输入要访问的网址（假设是 www.sohu.com）。然后按下键盘的 Enter 键，将获取网页的任务交给浏览器去完成。

（2）浏览器的工作：浏览器首先要去寻找目标网站的 IP 地址。这个 IP 地址就好比我们要去某餐馆吃饭，那个餐馆的门牌号码。让我们记住门牌号码是非常困难的，特别是在好吃的饭馆很多的情况下。因此我们往往记住的是饭馆的名字，一般来说，名字比门牌号码更容易记忆，字数也没有那么长。在网络世界中，网址就相当于饭馆的名字，而 IP 地址则是饭馆的门牌号码。

【网址与 IP 地址的关系】

网址和 IP 地址是对应的，这种对应关系由一种名为 DNS 的协议和系统来记录。浏览器在接到访问网址的请求后，首先查询 DNS 服务器，利用如上的对应关系，把该网址翻译成某个具体的 IP 地址，再向绑定该 IP 地址的某台服务器请求相关网页文件并显示在窗口中，从而完成网页浏览的整个过程。

（3）网站处理浏览器发来的用户请求，并给予反馈：经过前面的两个步骤，用户的请求就被传递到了网站的门口，在服务器上运行的网页发布服务首先向用户返回一个预先设置好的首页面。这个首页面就好比饭馆中的菜单一样，上面列出了网站都有哪些栏目，然后用户就可以根据自己的喜好，进入各个频道浏览了（很像点菜的过程）。假设一个用户非常喜欢看娱乐新闻，就可以单击首页上的娱乐新闻链接。网页发布服务就像一个服务员一样，得到了该用户的请求，将需要的网页发送回去，这样就完成了一次如图 3-11 所示的请求与反馈的过程。

在整个过程中，用户所使用的浏览器可能不同，服务器上运行的网页发布服务也可能不同，那么，浏览器和网页发布服务是靠什么来沟通的呢？答案就是协议。

3.1.7　协议

所谓协议，英文叫做 Protocol，是一组在网络上发送和接收信息的规则与约定。这些规则控制在网络设备和程序之间交换消息的内容、格式、定时、顺序以及错误控制。通俗地讲，协议就是不同网络设备、程序之间交流沟通的语言。

【协议与方言】

中国有很多种方言，如果不特别学习，各个地方的人相互交流就可能存在一些障碍，你听不懂我说的，我也听不懂你说的。因此，有必要制定一种大家都能够听懂的中国话的标准，这就是普通话产生的原因。普通话也可以说是一种协议。

前文提到的 FTP，与本节提到的 HTTP 都是协议的名称，是为了应用在不同的场合下方便程序之间交流的标准。

浏览网站正是由很多个请求——反馈这样的过程组成的。在一个用户浏览的同时，其他用户也可以同时浏览该网站，正如饭店里不止一张桌子的人在吃饭。能够同时容纳很多人进行网络浏览而不“手忙脚乱”，这是考验网页服务程序的一个重要指标之一。当然，人再忙也有个限度，如果访问量太大，就得增加饭店的服务员，扩大饭店的营业面积来应对。也就是说，如果网站遇到类似的情况，为了达到同样的服务质量，就需要增加服务器的数量和质量，以及提高服务器的性能来实现了。

【HTTP 协议】

前文提到了 HTTP 协议，这里多介绍一下。实际上，互联网上的核心之一就是 HTTP 协议。它在构成互联网的各个部分：客户端、服务器中都得到实现，这样才能保证信息沟通、交流的进行。HTTP 协议有一些很复杂的定义，对行走在网络上的消息文本的结构以及客户端、服务器如何交换消息的方式、方法进行了规定。具体内容，读者可以参阅有关互联网、TCP/IP 的书籍。

3.1.8 前台页面与后台数据库

服务很重要。本节依然用饭馆来类比，简单说，一般的网站实际上都是由前台页面和后台数据库组成的。

1. 前台页面

前台页面实际上就是网站的最终用户在浏览器中看到的页面，相当于我们在饭馆中就餐的场所。不言而喻，前台页面的地位绝对重要：

- ❑ 前台页面上的各个元素要干净、整洁，正如餐馆就餐区的环境卫生。
- ❑ 前台页面的全部显示要尽可能快速，正如餐馆服务员上菜的速度。
- ❑ 前台页面实现的功能要满足用户的需求，正如餐馆服务员对于客户要求的满足程度。

本书所讲述的网站性能，很重要的一个指标就是上面的第二点，前台页面的显示速度。

2. 后台数据库

餐馆光有就餐场所是不行的，必须有一个强大的厨房、库房和采购部门的支持。这个后勤部门在网站上就是后台数据库的作用。后台数据库为前台页面提供了：

- ❑ 前台页面丰富多彩的内容，正如厨房为就餐者制作精美的菜肴。
- ❑ 保存用户在前台页面上操作的过程、结果，正如餐馆服务员常问的“菜有没有忌口”等类似的话，厨房再根据这些用户的偏好进行调整。有些高级餐馆甚至会记录并保留客户的更多信息（比如生日和口味、菜系、食材的偏好等），就餐者下次再来就会享受更贴心的服务。

3. 前、后台的物理位置

前台页面和后台数据库出于性能的考虑，一般都不被安装在一台服务器中。这其实很

容易理解：在餐馆中，厨房和就餐区域一般来说是分开的（餐饮术语分别为“前场”和“后场”，可见使用了类似的称谓方法），如果在一起，厨师做菜、服务员上菜难免会磕磕碰碰，乱作一团。

后台数据库的这两个功能是由以下两个子部分来实现的：

- 一系列供网站内部编辑人员增加、删除、更新数据的程序或者网页，一般称之为后台管理程序。这部分相当于厨师的作用。
- 一个或者多个数据库：用于保存网站的内容、后台管理程序与前台用户操作的结果。相当于餐馆的库房。

3.2　Web 开发技术简介

还记得小白吗？在经过短短几天的学习之后，他已经有了一些测试的经验。虽然目前他需要做的只是黑盒测试，但为了尽快熟悉被测试的网站，还是很希望学习一下目前主流的开发网站的方法。本节就简单介绍目前流行的几种 Web 开发技术。

- JavaWeb 开发技术。
- .NET 开发技术。
- PHP 开发技术与其他开源平台。
- AJAX 开发技术。

【AJAX 的特别之处】

最后一种 AJAX 开发技术与其他 3 种并不属于同一种范畴，严格说来其更类似一个具体的技术实现方法，可被包含于前面 3 种开发技术的任何一个里。AJAX 技术的简单原理将在 3.2.5 节介绍。

3.2.1　Java 简史

本节简单介绍 Java 平台的 Web 开发技术总体特点。Java 是一种简单的、面向对象的、分布式的、解释型、支持多种操作系统、多线程的动态编程语言。1995 年，由大名鼎鼎的 SUN 公司推出，之后不久便成为主流的 Web 开发技术。

【Java 原名 OAK】

Java 语言最早诞生于 1991 年，起初被称为 OAK 语言，是 SUN 公司为一些消费性电子产品（类似现在使用的手机等）而设计的一个通用环境。他们最初的目的只是为了开发一种独立于平台的软件技术，而且在网络出现之前，OAK 可以说是默默无闻，甚至差点夭折。但是，网络的出现改变了 OAK 的命运。

在 Java 出现以前，Internet 上的信息内容都是一些乏味死板的 HTML 文档。这对于那些迷恋于 Web 浏览的人们来说简直不可容忍。他们迫切希望能在网络中看到一些交互式的内容，开发人员也极希望能够在 Web 上创建一类无需考虑软硬件平台就可以执行的应用程序，当然这些程序还要有极大的安全保障。对于用户的这种要求，传统的编程语言显得无能为力，而 SUN 的工程师敏锐地察觉到了这一点，从 1994 年起，他们开始将 OAK 技术应用于 Web 上，并在第二年正式以 Java 这个本意为爪哇岛或者咖啡的名字来命名。

3.2.2　Java 语言的特点

Java 主要有平台无关性、面向对象、安全性、分布式等几个特点。其中，和 Web 开发有相对更紧密关系的是下面两个特点。

1. 平台无关性

平台无关性是指 Java 能运行于不同的平台。Java 引进虚拟机原理，并运行于虚拟机，实现不同平台的 Java 接口之间。使用 Java 编写的程序能在世界范围内共享。Java 的数据类型与机器无关，Java 虚拟机（Java Virtual Machine）是建立在硬件和操作系统之上，实现 Java 二进制代码的解释执行功能，提供于不同平台的接口。

电子商务要求程序代码具有基本的要求：安全、可靠、同时要求能与运行于不同平台的机器的全世界客户开展业务。Java 以其强安全性、平台无关性、硬件结构无关性、语言简洁同时面向对象，在网络编程语言中占据无可比拟的优势，成为实现众多电子商务系统的首选语言。

如图 3-12 显示了某网站的首页。对于使用者来说，如果在浏览器的地址栏中出现 jsp 或者 servlets 等字样，就可以判断出当前系统采用了 Java 平台的 Web 开发技术。



图 3-12　应用 Java、Jsp 技术的某网站首页

2. 分布式

Java 建立在扩展 TCP/IP 网络平台上。库函数提供了用 HTTP 和 FTP 协议传送和接受信息的方法。这使得程序员使用网络上的文件和使用本机文件一样容易。

代码 3-1 显示了一个完成判断传入页面参数 username 是否为空的 JSP 代码。

代码 3-1　判断传入参数是否为空的 JSP 代码

```
<%@ page import="hello.NameHandler" %>
<jsp:useBean id="mybean" scope="page" class="hello.NameHandler" />
```

```
                                                    // Java 的 Bean 名称
<jsp:setProperty name="mybean" property="*" />
<html>
<head><title>Hello, User</title></head>
<body bgcolor="#ffffff" background="background.gif">
<%@ include file="dukebanner.html" %>               // JSP 中包含文件的写法
<table border="0" width="700">                      //页面的表格设计
<tr>
<td width="150">   </td>
<td width="550">
<h1>My name is Duke. What's yours?</h1>
</td>
</tr>
<tr>
<td width="150"   </td>
<td width="550">
<form method="get">
<input type="text" name="username" size="25">
<br>
<input type="submit" value="Submit">
<input type="reset" value="Reset">
</td>
</tr>
</form>
</table>
// 下面判断传入参数是否为空，若不为空则定向到另外的页面
<%
   if ( request.getParameter("username") != null ) {
%>
<%@ include file="response.jsp" %>
<%
   }
%>
</body>
</html>
```

注意在页面中为了与 HTML 标签相区别，JSP 代码用<%和%>包括起来。

3.2.3　.NET 平台的 Web 开发技术

.NET 平台是由微软推出的可用于 Web 开发的技术。和 Java 平台类似，但它绝大部分情况下只能在 Windows 平台下运行。.NET 平台与 Java 平台类似，在程序运行所在的机器上都要安装一个运行库，称为.NET Framework。至笔者截稿为止，最新版本为 3.5sp1，但是 4.0 版本已经呼之欲出。

在 Web 开发方面，.NET 平台提供了服务器端网页文件，其后缀名为 aspx。另外，在.NET 平台推出之前，微软的 Web 开发解决方案主要采用后缀名为 asp 的服务器端脚本技术来实现，目前该项技术依然有一些使用者。如图 3-13 显示了 IE 7 中某网页的效果，从地址栏中的后缀名即可判断出来它采用 aspx 编写。代码 3-2 则是读取并显示某文件属性的 aspx 代码。

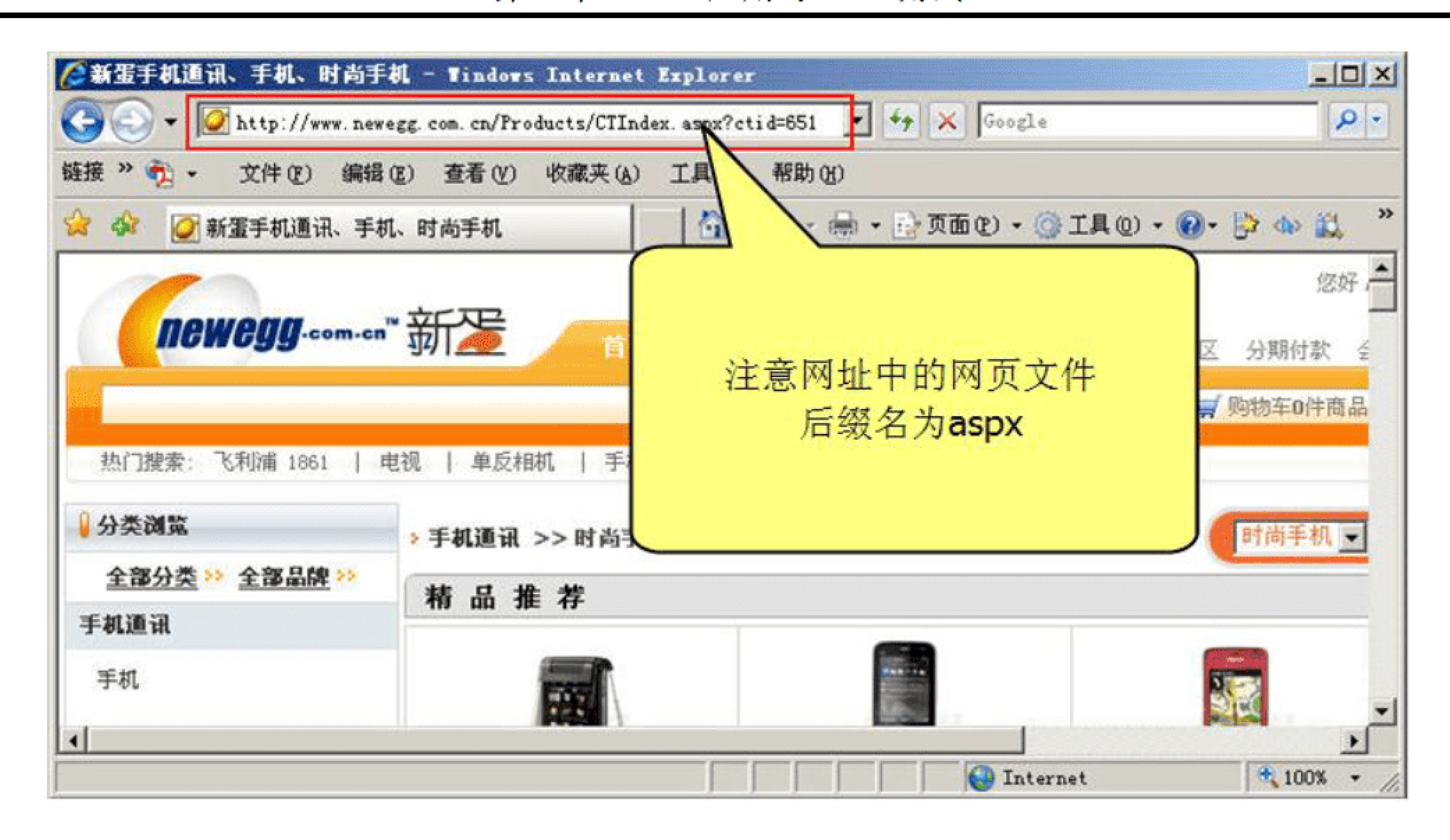


图 3-13　用.NET 技术开发的一个网页

代码 3-2　读取并显示文件属性的 aspx 代码

```
<% @Page Language="C#" Debug="true"%>
<% @Import Namespace="System.IO" %>            //引入 System.IO 命名空间
<html>
<head><title>File Info</title></head>
<body>
<%
string strFile2Show = Request.QueryString.Get("file");//传入文件名作为参数
File thisOne = new File(strFile2Show);              //取得该文件对象
%>
<table>
<tr><td>文件名：</td><td><%=thisOne.Name%></td></tr>
<tr><td>全名：</td><td><%=thisOne.FullName%></td></tr>
<tr><td> 文 件 创 建 日 期 :</td><td><%=thisOne.CreationTime.ToString()%>
</td></tr>
</table>
</body>
</html>
```

【CodeFile 方式存放代码】

aspx 页面代码既可以存在于网页本身之中，也可以单独成为代码文件。前者即是动态网页的传统方式，与 JSP 等类似，用<%和%>标签将代码与 HTML 标签隔离开来。后者则是 CodeFile 方式，将代码保存在单独的 aspx.cs 文件中，在网页头部用 CodeFile 属性指向这个代码文件。这样做的好处是页面表现与具体逻辑分开，另外代码文件被编译，比直接写在网页上的直译程序快，还可以防止程序代码外泄。

3.2.4　基于 PHP 的 Web 开发技术

PHP 和以上两种平台不同，它是开放源代码的。一般来说，在实际的开发和应用中，PHP 一般和开源数据库 MySQL 搭配使用。

图 3-14 是 IE 7 中显示的利用 PHP 技术开发的某网页，代码 3-3 则显示了一个完成读取数据库功能的 PHP 页面代码。

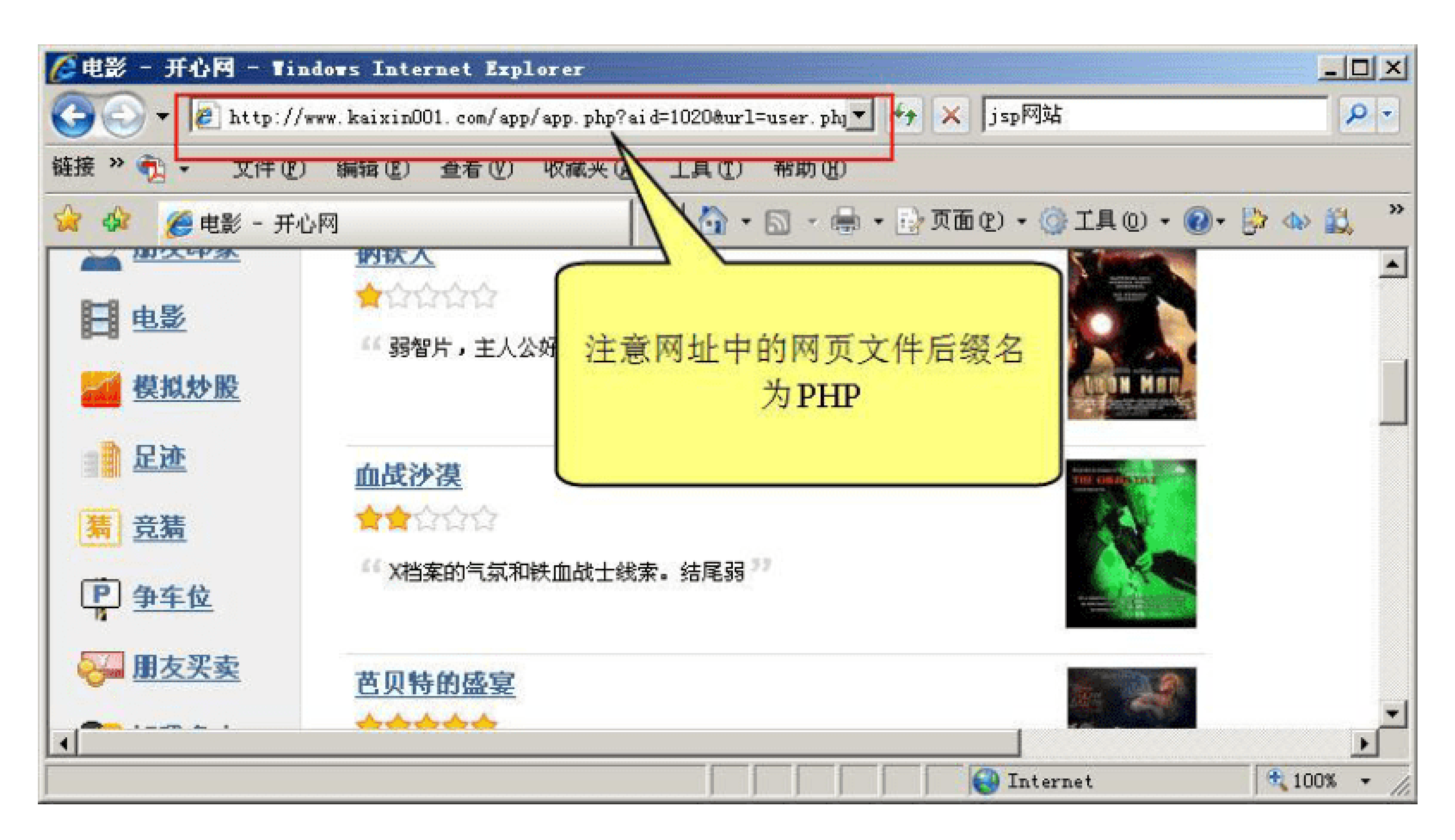


图 3-14　利用 PHP 技术实现的一个网页

代码 3-3　对数据库进行查询的 PHP 代码

```
<?php
#这是 PHP 的注释符号。下面一行是包含文件的写法
include("globalenv.php");
#判断传入参数是否存在
if(array_key_exists('user',$_GET)!=false){
#若参数存在，判断是否为空值
    if($_GET['user']!=NULL){
    #取得参数值
    $user=$_GET['user'];
    #数据库连接
    mysql_connect('localhost:'.$port,$username,$password);
    @mysql_select_db($database) or die("Error:Unable to select the data
    base");
    #获取当前时间
    $now=strftime('%Y-%m-%d');
    #计算出距今 7 天前的时间
    $lastWeek=date("Y-m-d",(strtotime("last week",time())));
    #查询 SQL 语句的拼接
    $query="SELECT * from play where user='".$user."' and dateCreated
    between '".$lastWeek."' and '".$now."' order by dateCreated desc";
    #包含其他文件
    include("querydb.php");
    }
}
?>
```

注意其中 PHP 代码的分隔符是<?php 和?>，当然，也可以修改为<%与%>，不过要额外设置。

【测试工程师的选择】

前面介绍的 3 种 Web 开发技术三足鼎立，代表了目前网站开发的 3 种主流平台，他们都各有千秋，有自己固定的“粉丝”阵营。从以上几张图片以及代码片段来看，3 种平台技术实现的页面在外观、代码上并没有太多本质不同，另外，Web 应用一般只会采用其中

一种平台进行开发，因此对于测试工程师来说，如果主要进行黑盒测试，并不需要对以上 3 种技术的具体编程技巧都非常了解，只需要根据当前被测试 Web 应用的实际情况，尽量多熟悉这种技术即可。

3.2.5　AJAX 开发技术

AJAX 开发技术与以上 3 种 Web 开发平台并不属于同类范畴，它可以说是一种处理客户端与服务器交互的新方法，因此 3 种平台都可以应用。

【AJAX 的设计思想】

AJAX 技术的主要思想就是在不用页面刷新的情况下在网页后台提交信息给服务器并返回结果。由于省去了页面刷新的过程，所以 AJAX 页面只改变需要改变的内容部分，其他网页元素不再重复装入浏览器，省去了从服务器传输这些数据的时间和字节数，从而使得用户感觉到的页面响应时间变短，改善了用户体验。

图 3-15 和图 3-16 分别显示了在利用 AJAX 技术的某网站中，提交评论之前和提交过程中的界面，可以发现页面并没有再次刷新。

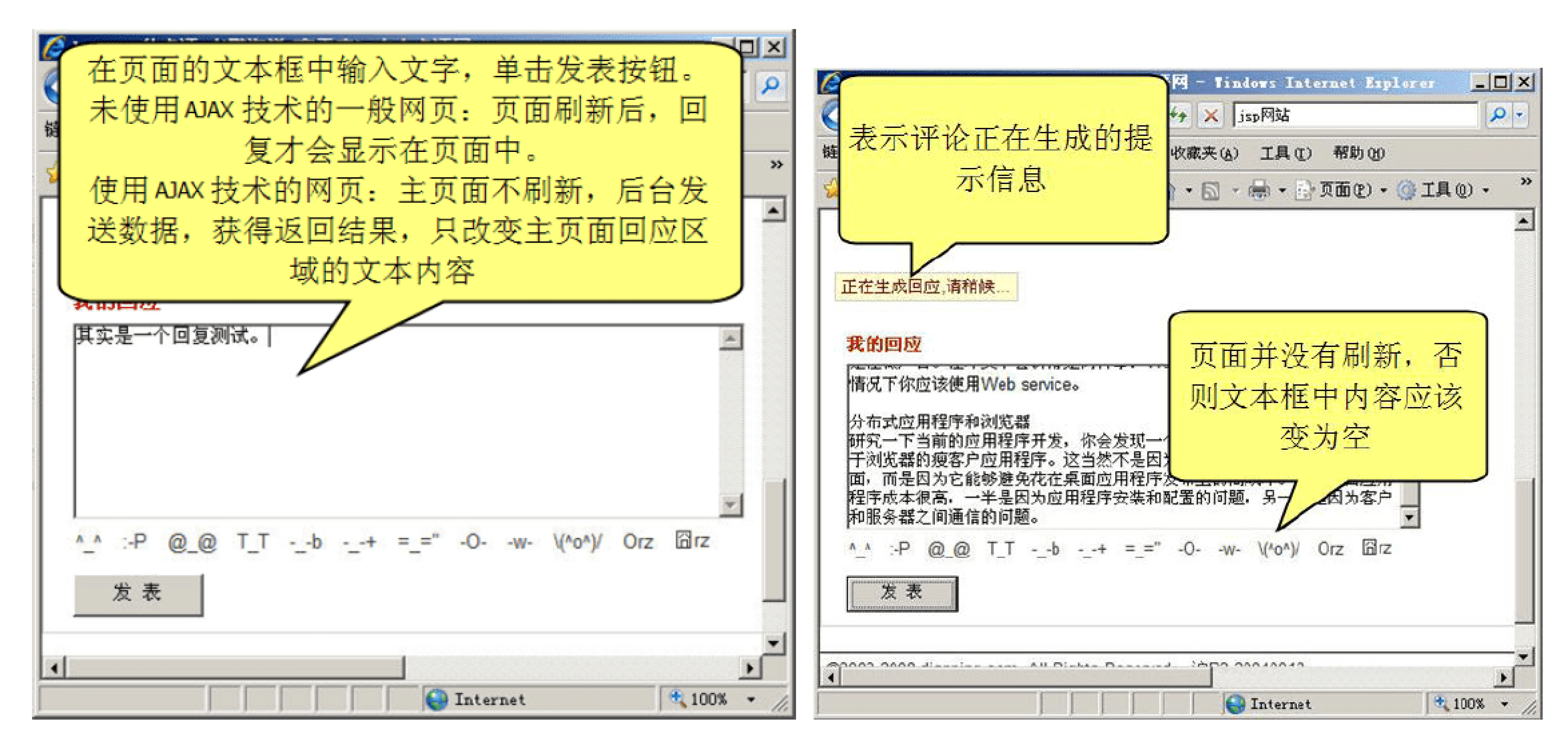


图 3-15　利用 AJAX 技术使得提交回复不刷新当前页面　　图 3-16　回复提交过程中页面并没有刷新

在简单介绍了以上这几种技术之后，相信读者已经对当今 Web 应用的主流技术有了粗浅的感性认识。因此，3.3 节将回归到测试工作上来，讲述 Web 测试的内容和功能测试的特点与方法。

3.3　Web 功能测试的特点与方法

3.2 节概述了 Web 开发的几种主流平台，还介绍了 AJAX 技术。对于具体技术有一定程度的了解是 Web 应用测试工程师应该具备的，因为它将非常有利于在工作中理解软件实现、开展白盒测试。不过，由于黑盒性能测试是本书的主要内容，笔者还是将被测试的网

页作为黑盒来处理，不会过多考虑它背后使用了何种技术来实现。

本节是本章的重点，通过举例概要讲述 Web 测试的方法。在本节的开头，首先提出一个问题，请读者想一想，如果自己是小白，面对一个全新的网站，该如何着手去测试呢？

这个问题没有什么固定答案，但是，有一个答案可以说是很自然就能想到的：把自己所能看到的网页各个部分都“测试”，或者说单击一遍。之所以把测试这个词加上了引号，是因为目前这种操作方式更类似对网页的试用，不具备系统化、计划性的特点。

下面就从这个肤浅的出发点开始。

3.3.1　网页测试的组成部分

如图 3-17 是某网站一个典型的网页，首先观察一下网页中包含了哪些内容，一旦确定了这些内容，就可以对每一项分别展开测试。



图 3-17　一个典型的网页

在图 3-17 中，我们发现了文本框、下拉菜单、表单提交按钮（即查找按钮）、各类图片、文字链接，实现网页语言选择的链接（有 English、繁体中文等多个）等。因此，需要测试这些元素显示是否正常，功能是否正常，这就是网页的功能测试部分。

具体而言，网页的功能测试需要验证以下方面：

- 文字或者图片链接的有效性测试，简称链接测试。
- 表单提交测试，简称表单测试。
- 网页内容、用户界面和多语言测试。
- 浏览器交互测试，比如修改浏览器选项设置，以发现当前网页可能存在的问题。
- 其他网站特定功能的测试：比如上传功能等。

除了功能测试之外，网页测试还要考虑如下一些问题：

- ❑ 性能测试（包括负载/压力测试）：本书讲述的主要内容。
- ❑ 兼容性测试：对各个厂商的浏览器，以及其不同版本之间网页显示是否正常的测试。
- ❑ 安全测试：Web 应用是否构成安全隐患，比如造成密码泄露、数据库被侵入等的风险。

由于性能测试是本书的重点，将在后面的章节陆续讲到，这里就不叙述了。对于功能测试，由于内容相对其余的几种要多些，将在 3.3.2 节中做具体的讲解。至于兼容性测试、安全测试，将在本章的后面内容中再做介绍。

3.3.2　链接测试及其要点

本节讲述链接测试以及过程中需要注意的问题。

1．链接测试的目的

链接测试验证网页上的所有链接，目前包括文字、图片（静止图片和动画图片）和 Flash 动画 3 种，是否都指向正确的、真实存在的网页或者其他位置（比如下载的文件、网页的一个锚点）。需要注意的是，验证链接本身格式、字体大小等是否符合设计说明则属于后面要讲述的网页用户界面测试，将在随后讲到。

具体而言，做链接测试一般都要对表 3-1 中列出的 3 个问题进行验证。

表 3-1　链接测试需要关注的问题

问　　题	出现错误时候的一般表现
链接指向是否正确？是否指向了正确的网页？	打开了其他的网页
指向的网页是否存在？	页面找不到的 404 错误
链接的打开方式符合设计说明吗？如果是利用 JavaScript 等脚本打开新网页，能否成功实现？ （链接的打开方式指的是单击链接后，新开页面还是在当前页面跳转。）	错误的新网页打开方式。单击 JavaScript 后浏览器无反应 （注意：如果浏览器禁用了弹出窗口，浏览器会出现提示。）

2．不同链接的测试要点

对于文字链接，没有特别之处。对于图片链接，验证要复杂一些，要考虑以下的特殊情况：

- ❑ 有的图片链接是以图片地图的形式存在的，即一幅图片，不同的部位指向不同的目标网页，多用于导航条、网站地图等场合。如图 3-18 是 Yahoo 网站上各国家或者地区分站的入口。虽然是同一个世界地图的图片，但单击不同的蓝点会切换到不同的页面。
- ❑ 有的图片是动画 Gif 格式或者直接就是 Flash 动画，这时候要注意验证动画的每一帧（即每一次图像内容变化）所链接的网页是否正确。容易被忽视的恰恰是它们第 2 帧之后的链接，因为测试工程师往往在网页显示完毕之后就测试各个链接，这时候动画图片后几帧尚未显示。不同帧对应不同链接的动画图片使用较少，但在首页的新闻提示栏或者各个友情链接被合并成一个动画图片的时候，还是能够

碰到的。

- 对于以上这几种图片或者 Flash 动画，只能把它们当成多个链接来处理，一个一个地进行验证，确保测试完全。

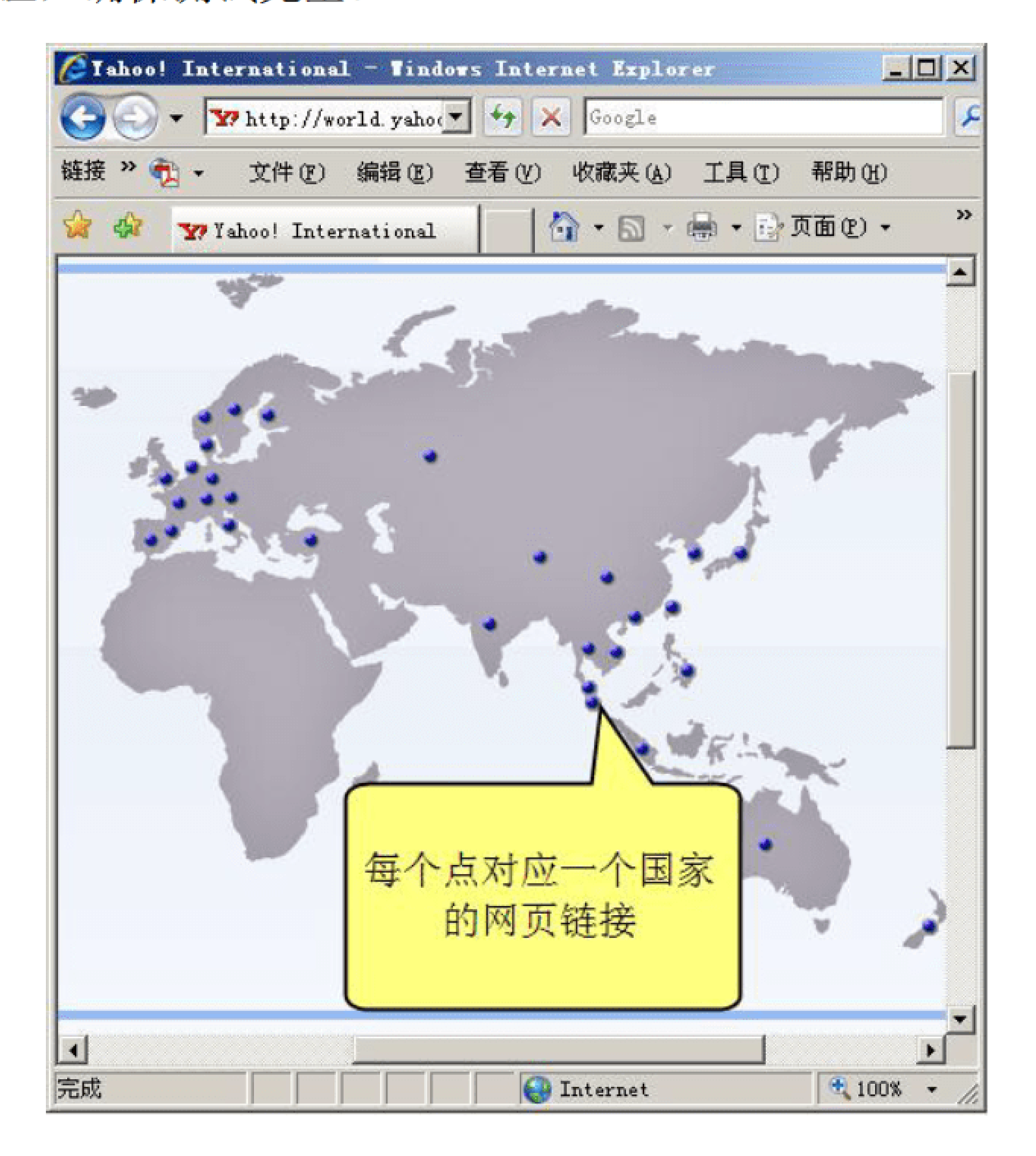


图 3-18　Yahoo 的图片地图导航

3．链接测试方式的选择

由于网页上的链接一般都非常多，手工地逐个验证是非常容易遗漏和出现错误的。这时候可以考虑利用自动测试的方法：

- 利用某些链接验证工具软件进行网页的链接测试。
- 利用某些测试工具软件的脚本录制功能，将用户单击链接的行为录制下来进行验证。
- 利用 HTML 等知识，通过验证 DOM 里<a>标签的值进行测试。

三种方法都需要较多的测试准备工作，第二种方法的编程量可能会更多一些。因此，在实际使用中，要综合考虑进行方法的取舍。

【手工或自动链接测试的选择】

如果被测试的链接经常改动，无法预知，不妨直接采用手工测试链接的方式。反之，如果各个网页版本某链接都比较固定，可以将它的验证变为自动，这样可以省去大量的时间。

3.3.3　链接测试工具 Sleuth

在前文中，我们介绍了可以用某些链接验证工具软件达到测试的目的。这里简单介绍

一种免费链接验证工具 Xenu's link sleuth，简称 Xenu 的使用。

所谓 Sleuth，英文本意是指动物的足迹，用来形容链接测试，发现哪里链接断了（相当于痕迹没有了），还是很贴切的。该软件可以在 http://home.snafu.de/tilman/xenulink.html 下载到，是一个压缩包，解压缩后软件不大，几兆而已。

压缩包中只有一个 Setup.exe，运行它，安装 Xenu 软件。安装完毕后，运行该软件，主界面如图 3-19 所示。

Xenu 软件的主要功能都是通过图 3-19 中高亮显示的两个菜单命令来完成的，只需要分别单击这两个命令，在其中输入待检测的网址，即可开始测试过程。测试结束后，根据提示确定即可生成链接检测的报告，如图 3-20 所示。

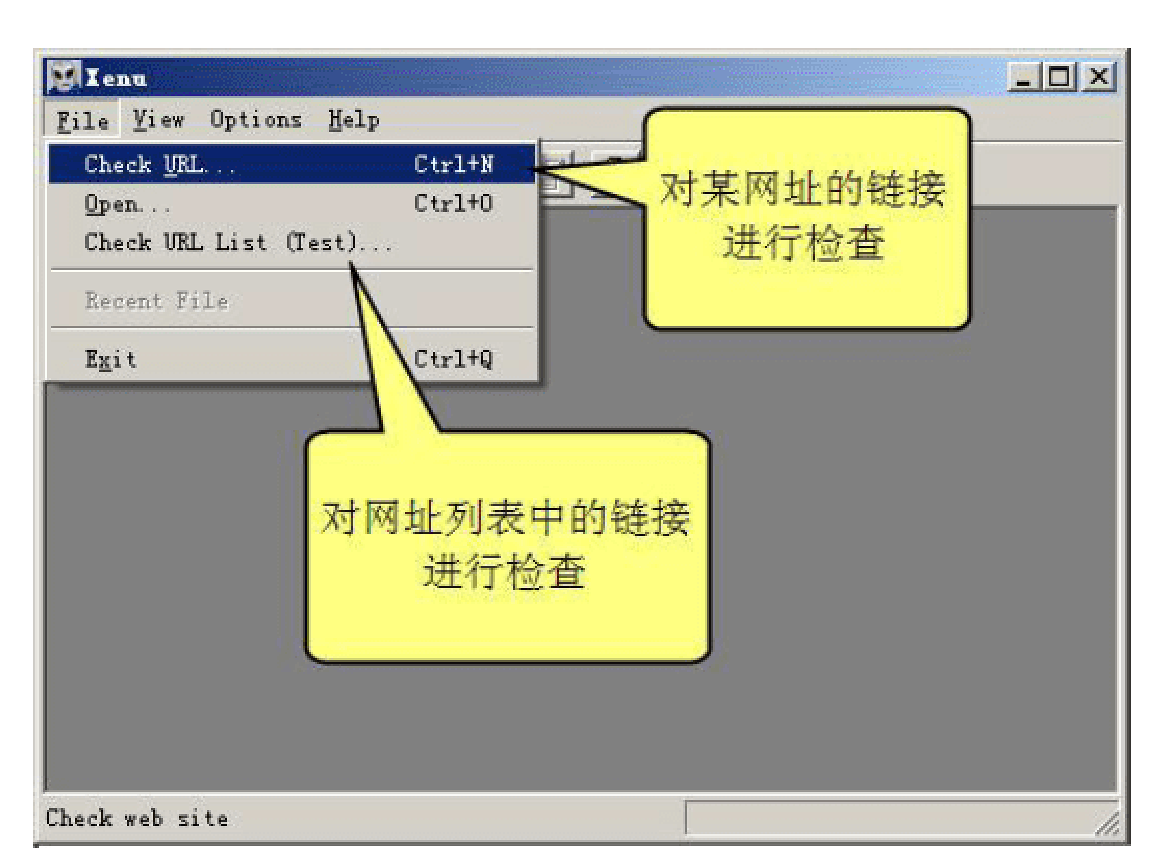


图 3-19　Xenu 的软件运行界面

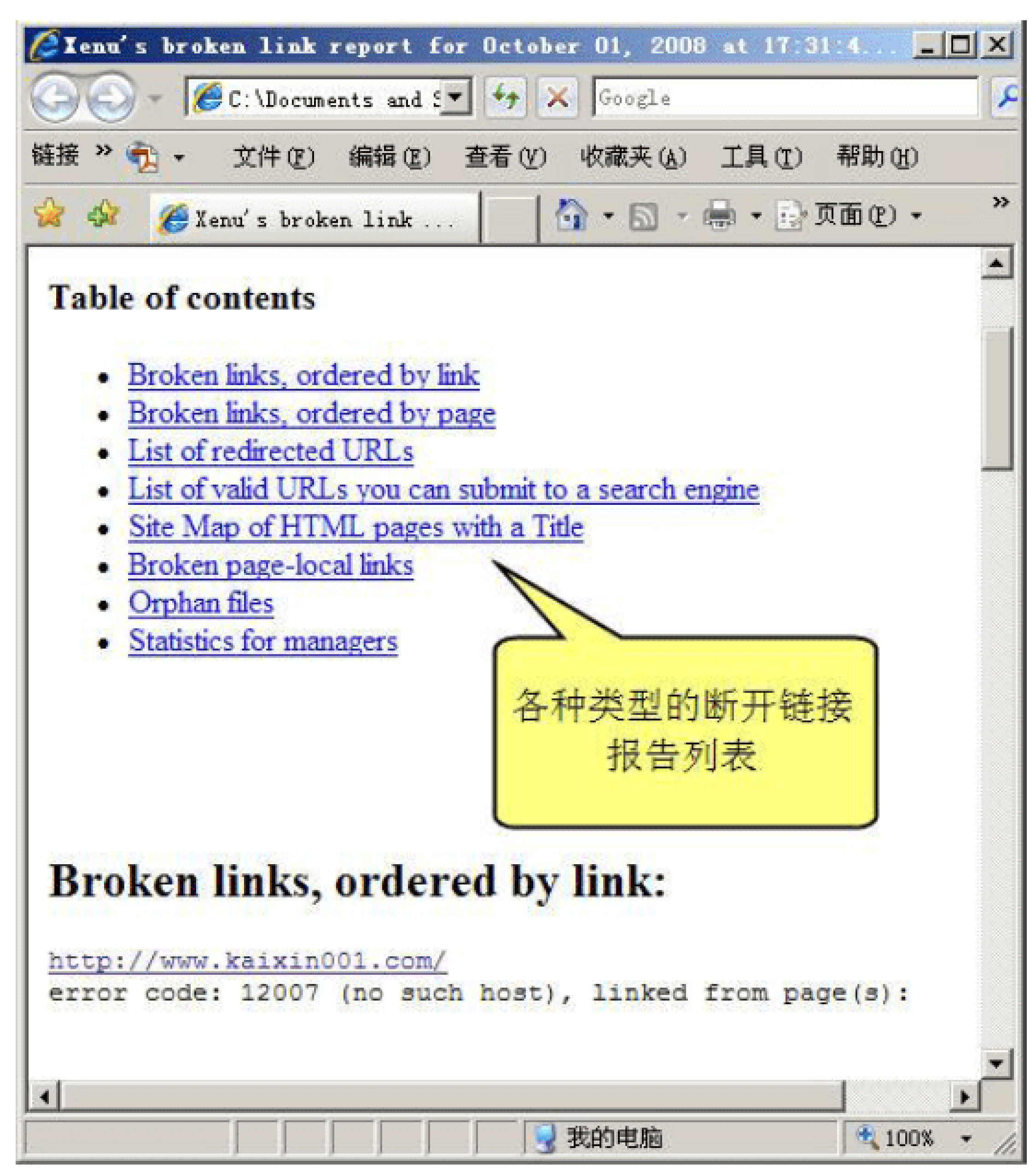


图 3-20　Xenu 针对某网址生成的链接验证报告

与另外的链接测试软件相比，Sleuth 有如下优点：

- 界面简单易用。
- 可以反复验证链接（对于偶然的网络错误导致链接不可访问是很有用的，增加了可靠性）。
- 可以发送邮件报告（可以让它晚上不上班的时候去测试，测试人员上班后查看邮件即可）。
- 支持安全套接层 SSL 的网站（也就是包含 https://开头那样的网页，用于安全性要求比较高的场合，比如网上支付等）。
- 能够支持互相跳转网页上链接的验证。

【W3C 的官方工具】

此外，互联网标准组织 W3C 的官方网站上也有一个网页用于链接检查，网址是 http://validator.w3.org/checklink。它可以用于简单的链接不很多的网页验证，适于对网站的链接情况做一个大致的了解，因为并不需要软件的安装，使用快捷。

3.3.4　孤儿网页

和链接有关的还有一种所谓的孤儿网页，英文叫做 Orphan page。它是指在网站中的某个网页没有被其他任何网页上的链接联系到，像没有亲人的孤儿一样，故而得名。孤儿网页只有在用户完全知道页面的全部地址才能被访问到。

【孤儿网页产生的原因】

孤儿网页产生的原因一般都是网站的页面不断修改、变化版本，导致之前开发后来被新页面替代的旧网页遗留在服务器上造成的。

既然没有别的链接能够指向孤儿网页，那么它留在服务器上也没有坏处吧？

回答虽然是没有很大的坏处，但容易产生 3 个方面的不利影响：

- ❑ 用户之前收藏了旧网页，并一直通过收藏夹直接访问该网页，造成用户获得信息得不到更新。这在获取网站的联系方式等页面上影响还是相当大的。
- ❑ 孤儿网页可能泄露一些网站的秘密信息，这样的孤儿网页一般是开发人员为了方便创建的，如果不在网站开发完成后及时删除，有可能泄露一些网站、服务器的安全信息，对网站造成潜在的风险。
- ❑ 孤儿网页增多会混淆网页功能，并占用服务器的空间。当然，现在硬盘和白菜一样便宜，损失几 KB、MB 硬盘空间并没有什么，但毕竟是一种浪费。另外，由网站旧版本文件演变形成的孤儿网页会干扰当前版本的网页，有时候会使得开发人员错误地修改了旧版网页，并将其替换为现有网页，导致网站功能出现异常。

因此，测试工程师有必要及时指出孤儿网页的存在及其位置。前文所讲述的 Xenu 软件也有发现孤儿网页的功能，如图 3-21 所示。

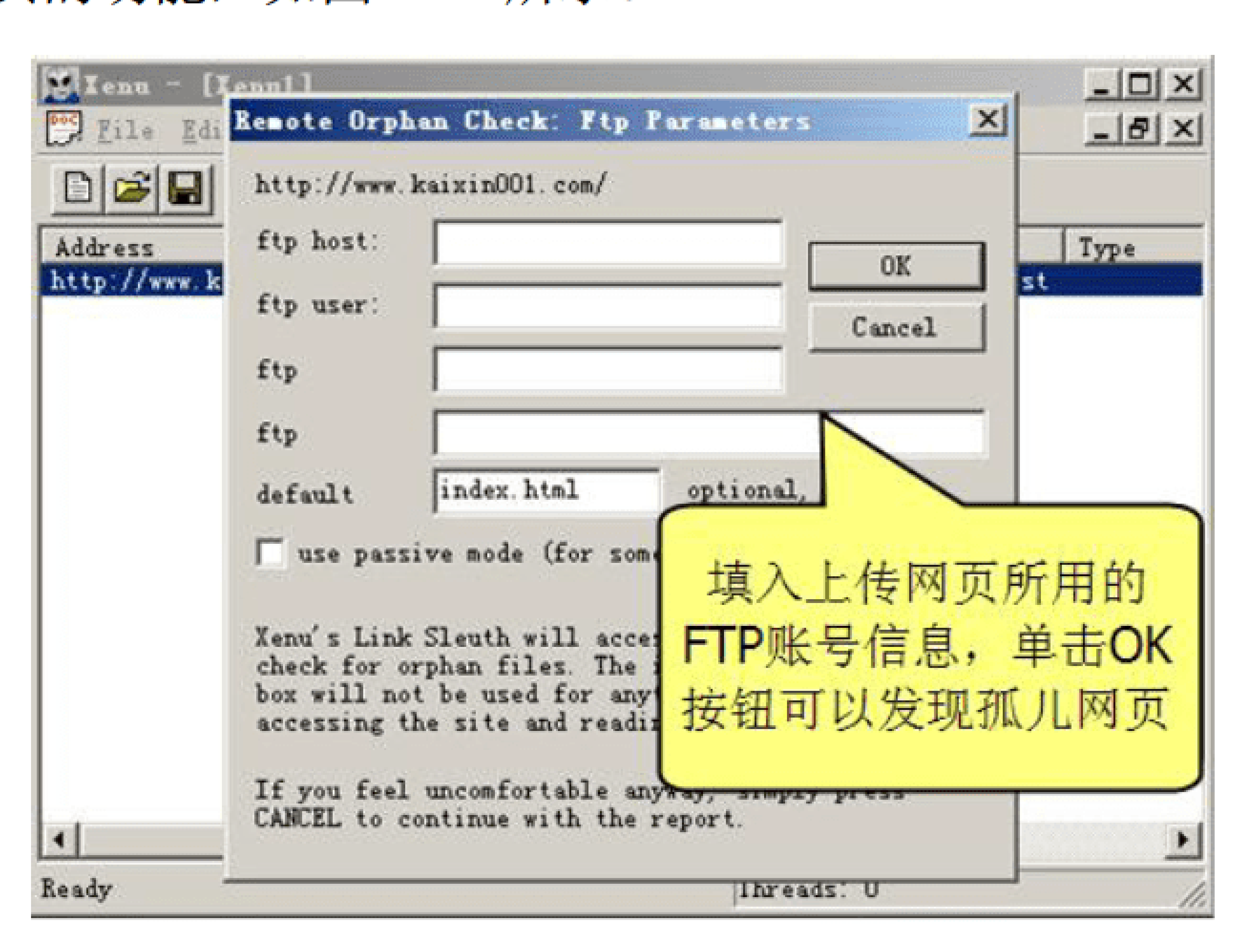


图 3-21　利用 Xenu 软件发现孤儿网页

其他不少的商业网站开发工具也都有这样的功能，比如业内常用的 Dreamweaver 等。

3.3.5　表单测试及其要点

表单是网页上用来接收用户输入的信息，并将结果反馈给用户的桥梁。表单由一系列

文本框、下拉菜单、选择框和提交按钮组成的。表单测试的目的就是验证网页上的表单能否正确提交信息，正常工作。

根据表单使用的场合和一般网页的设计要求，表单正常工作指的是：

当使用表单进行网站用户的在线注册、订单生成、修改的操作，以及用户反馈信息、搜索关键字的收集时，要确保提交按钮单击后能够提交全部需要的信息至服务器，这些信息的部分或者全部能够保存到后台数据库或者其他形式的文件中。表单提交完成后，应该给予用户成功或者失败或者是用户需要的列表等有意义的信息。

表单测试其实包含了很多的内容，根据实际操作的过程，在实际的测试工作中主要分为 3 个步骤：

（1）客户端表单信息的验证、收集和提交。

（2）服务器端用户信息的保存过程。

（3）服务器端提示信息的返回。

下面将分别介绍这 3 个过程的测试要点。

3.3.6　客户端表单信息的验证、收集和提交

这个过程，对于用户来说，就是在各种各样的输入框中输入信息，然后确认无误后单击提交按钮的过程。由于所有的操作都是在用户的机器上通过浏览器打开网页来完成的，所以称为客户端输入信息过程。

一般情况下，我们必须验证用户输入信息的有效性。这么做主要有两个原因：

（1）保证用户输入数据的有效，使得网站能够获取相对真实的用户信息。比如下面几种情况，都需要对数据进行验证。

- 对于网民来说，日期具备时效性，比如生日不可能是 1850 年 12 月 1 日。
- 对于网民来说，日期要符合规范，比如生日不可能是 1980 年 13 月 56 日。
- 对于目前的网上支付来说，信用卡号码必须符合发卡方的标准。
- 对于邮政编码、年龄等数据，根据需要，可以限制文本框中只能输入数字。
- 对于电子邮件来说，必须符合标准，简单地说，要具有@和.等符号。
- 对于备注、用户反馈类的信息，不可能让其填入很多的文字，要满足一定的字数限制。

（2）保证服务器端程序的安全性。有一种“注入”（Inject）攻击就是通过用户输入特别字符对服务器端数据库进行破坏。要在用户提交的信息中找到这些字符并过滤掉，这个过程可以在客户端进行，也可以在服务器端正式提交前进行。

【如何编写表单测试的用例】

总体说来，客户端表单信息的验证是比较重要的，Web 测试工程师需要利用前面章节介绍的划分等价类、边界值测试等进行充分的工作，保证不把不符合规范的数据提交到服务器端。

3.3.7　服务器端用户信息的保存过程

这个过程，一般是将用户信息经过处理保存到数据库或者文件中。在实际测试当中，

可以模拟一些客户端输入的数据，观察表单的行为。比如下面这些情况：

- 用户输入了合法的数据，但数据长度超过了数据库某字段的定义长度，也就是说存储数据的地方不够了，观察服务器端程序的行为和反馈。这样的情况，根据网站设计说明，一般有两种情况：直接报错和截取用户输入的可保存最大长度。对于前者方案的测试，需要修改客户端验证的文本长度限制或者服务器端数据库字段的定义长度；对于后者，则需要验证服务器端程序这部分代码的功能能否正常实现。
- 用户输入了客户端无法验证的数据，但这些数据对于服务器端是不合法的。
- 一些特殊情况的容错，比如断网等连接中断下，服务器端程序和数据库事务的处理情况等。

在这个测试过程中，模拟客户端输入的数据就是我们前面章节所说的测试用例。通过将不同的测试用例提交到服务器端，查看程序的行为，是 Web 测试的一般工作过程。

3.3.8　服务器端提示信息的返回

在服务器端程序处理完毕之后，无论结果是正确还是错误，都要给予用户以可阅读的反馈，避免出现过于简单的提示或者只有工程师才能读懂的错误代码。它的要求如下：

- 无论正确还是错误，都能返回到一个或者相应的结果页面。
- 对于正确的结果，提示信息要为用户着想，尽量完整。按照网站设计说明，对用户下一步的操作给出建议，或者过一段时间跳转到初始界面。
- 对于错误的结果，提示信息要列出一些可能的原因，同时表明表单提交的状态，并按照网站设计说明对用户下一步的操作给出建议。

在这个过程中，主要需测试的是信息的完备性，对用户是否友好。

3.3.9　网页内容测试

这部分主要是为了保证用户体验，使得网站对用户友好。它分为两大类：

- 网页内容测试：针对内容信息是否正确的测试。
- 网页用户界面测试：网页除去内容部分，针对各元素功能、页面风格等方面的测试。

【网页内容测试要点】

网页内容测试根据网站内部各部门对于页面内容的要求，检验 Web 相应页面提供信息的正确性、准确性、时效性，它们也是网页内容测试的目的。

具体而言，这 3 个要求的详细内容如下：

（1）所谓信息的正确性，是指网页上的信息是可靠的还是有谬误的。在前几年，曾经有这样一个案例，某个很大的在线购物网站标错了商品的价格，在很短的一段时间，就有不少用户订购了该商品。由于价格远远低于实际销售价格，销售商拒绝发货，最终网站和用户只好通过司法途径解决问题。这个案例充分体现了信息正确性的重要性。

（2）信息的准确性则主要针对专业术语，比如网站上的各项用户协议、在线支付的各种文本等都要符合相关规定和标准，要经过比较严格的检查，否则也会产生一些预想不到

的后果。

（3）信息的时效性主要保证信息的有效时间处于最近的时期内。这对于网站上，特别是首页的一些活动预告、广告宣传等是有用的。设想如果网站有一个嘉宾访谈类节目，但时间是去年某个时候举办的，当用户访问首页的时候，这个广告一直弹出来，很难设想用户看到后的反应。

有的人会问，这些都是编辑的工作，为什么要和 Web 测试工程师相关呢？测试工程师只测试和程序相关的内容。其实，这样理解是不对的。因为：

测试工程师负责测试整个网页。网页的内容也是网页的一部分，对用户体验有很大的影响。网页上的各个元素之所以存在，是为了用户更方便、更快捷、更舒服地浏览网页内容。

测试工程师有责任和义务发现问题。我们在前面的章节介绍过，错误的产生在所难免，比如，编辑在 Word 中写好的文字，可能由于其他人的感觉，觉得在网页中对齐不方便就增加几个字以保持美观。在这些情况下，测试工程师有责任、有义务尽早地发现问题，将修正 Bug 的成本降到比较低的水平。

3.3.10　网页用户界面测试

用户界面，英文名称叫做 User Interface，也简称 UI，是软件呈现给用户的外观部分。网页用户界面直接和用户体验相关，一般需要网站策划、网站内容负责人、网站美工、网页制作人员和网站测试工程师协同完成。测试工程师的工作一般是验证用户界面是否符合设计说明。

网页用户界面与表单测试不同，它是从大处着眼，一直到最小的页面元素，总体到具体的。主要测试内容包括：

（1）网页整体测试。这是指整个 Web 应用系统的页面结构设计，是给用户的一个整体感，主要考察页面的风格是否统一，易用性如何。测试时，可以考虑如下几个问题：

- ❑ 用户浏览网站的感觉是否舒适？
- ❑ 用户要查找的信息是否易于发现？
- ❑ 整个网站的设计风格是否一致？在切换栏目的时候是否有“跳出当前网站”的感觉？

（2）页面导航测试。网页上的导航条或者菜单列举出用户在网站内切换的便捷方法，主要由文字或者图片链接、跳转按钮、单独的 Div 浮动层、下拉列表和网页窗口等组成。在测试时需要考虑下列问题：

- ❑ 网站现有导航是否直观？
- ❑ 网站现有导航是否以首页或者各栏目的首页为出发点，能够做到来去自如？
- ❑ 网站现有导航是否足够，是否需要单独的页面列出站点地图、提供搜索框或者专门的帮助页面？

（3）页面样式表测试。这部分主要针对网页上各个元素的显示进行测试，比如：

- ❑ 图片的替换文本是否具备？图片的大小是否限制？
- ❑ 文本、按钮等元素的字体、颜色、边框是否符合要求？同一个页面没有特殊要求的文字字体是否一致？

- ❑ 网页各应用了样式风格的元素是否符合设计说明中的样式表设定？样式表的层次关系是否与设计保持统一？

对于用户界面测试，一般都是采用人工测试的方法，但是，验证部分重要文字的内容（比如按钮文字、重要链接等）、验证页面元素位置等可以通过自动测试的方法：利用 JavaScript 获取 HTML 或网页相应标签的 InnerText 值、Top 值等，将其与标准的参照值进行比较确定正确显示与否等方法来实现。

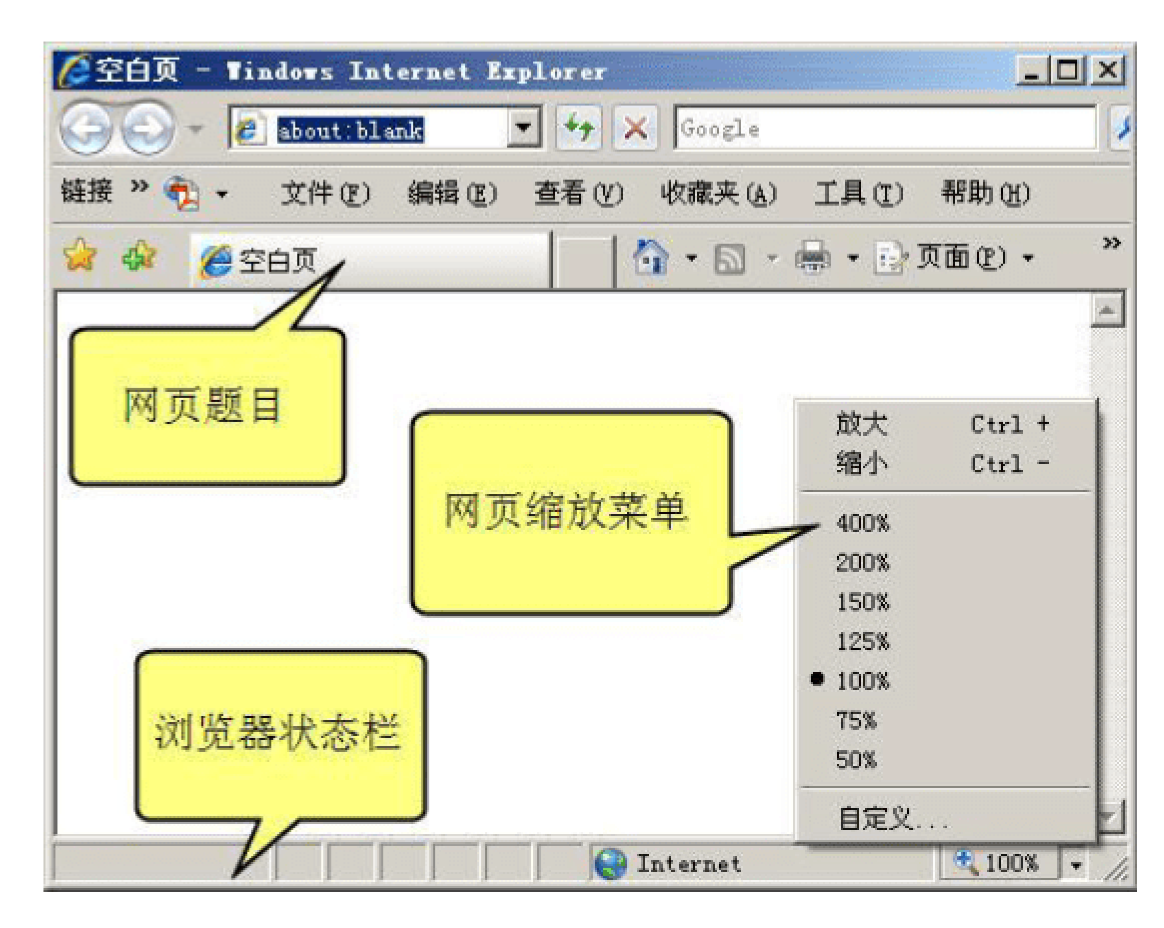
3.3.11　浏览器交互测试

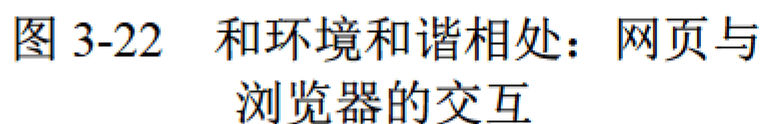
浏览器交互测试内容相对简单，主要考察网页与浏览器交互的部分功能是否正常，有如下几个问题需要考虑：

- ❑ 如果网页需要在浏览器状态栏中显示文字和一些特殊效果（比如部分网站采用的文字行进效果），确认它是否正常。
- ❑ 在浏览器中进行网页大小缩放比例的改变时，显示效果能够正常。
- ❑ 在浏览器中能否正确显示网页的题目。

以上 3 点的示意图如图 3-22 所示。

（1）在某些网站，浏览内容时需要安装特别的插件，比如 Flash 播放插件、数字版权插件、播放音视频的插件。这时候，浏览器会弹出一些信息加以提示，在 IE 浏览器中是一个黄条，称为“信息条”（Information Bar）。要给予用户足够的信息对这些插件进行说明，保证用户的使用，如图 3-23 所示。




图 3-22　和环境和谐相处：网页与浏览器的交互

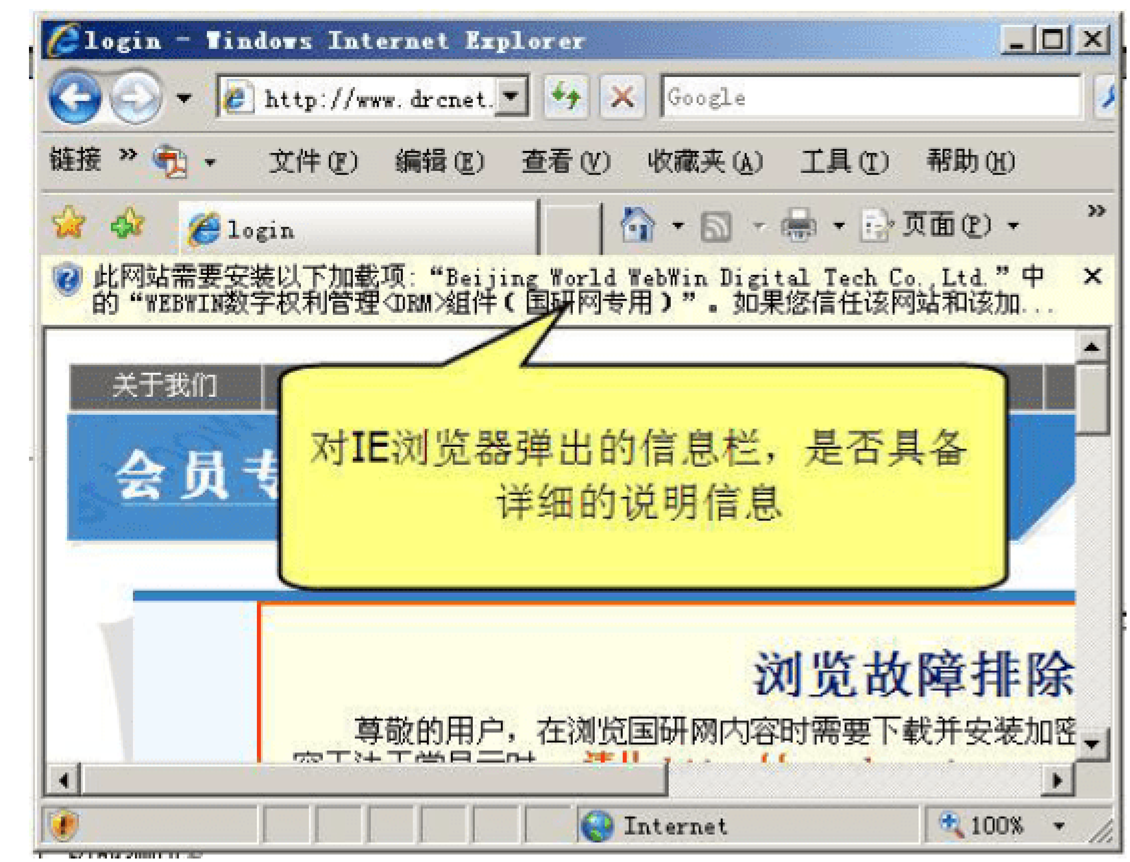


图 3-23　对浏览器提示安装的插件，要有足够的说明信息

（2）在浏览器中修改字体的大小，网页中字体的大小显示要正常：

- ❑ 对于应用有样式表的文字，字体应该忠实于样式表的设置，大小不变。
- ❑ 对于没有应用样式表的文字，字体应该服从浏览器的大小设置。

在 IE 中设置网页字体大小的菜单如图 3-24 所示。同样，对于其他浏览器也有类似设置。浏览器同时还可以禁用网页上应用的样式表，有时间的话也可以测试一下页面显示效果，以确保更多的用户能够正常浏览。

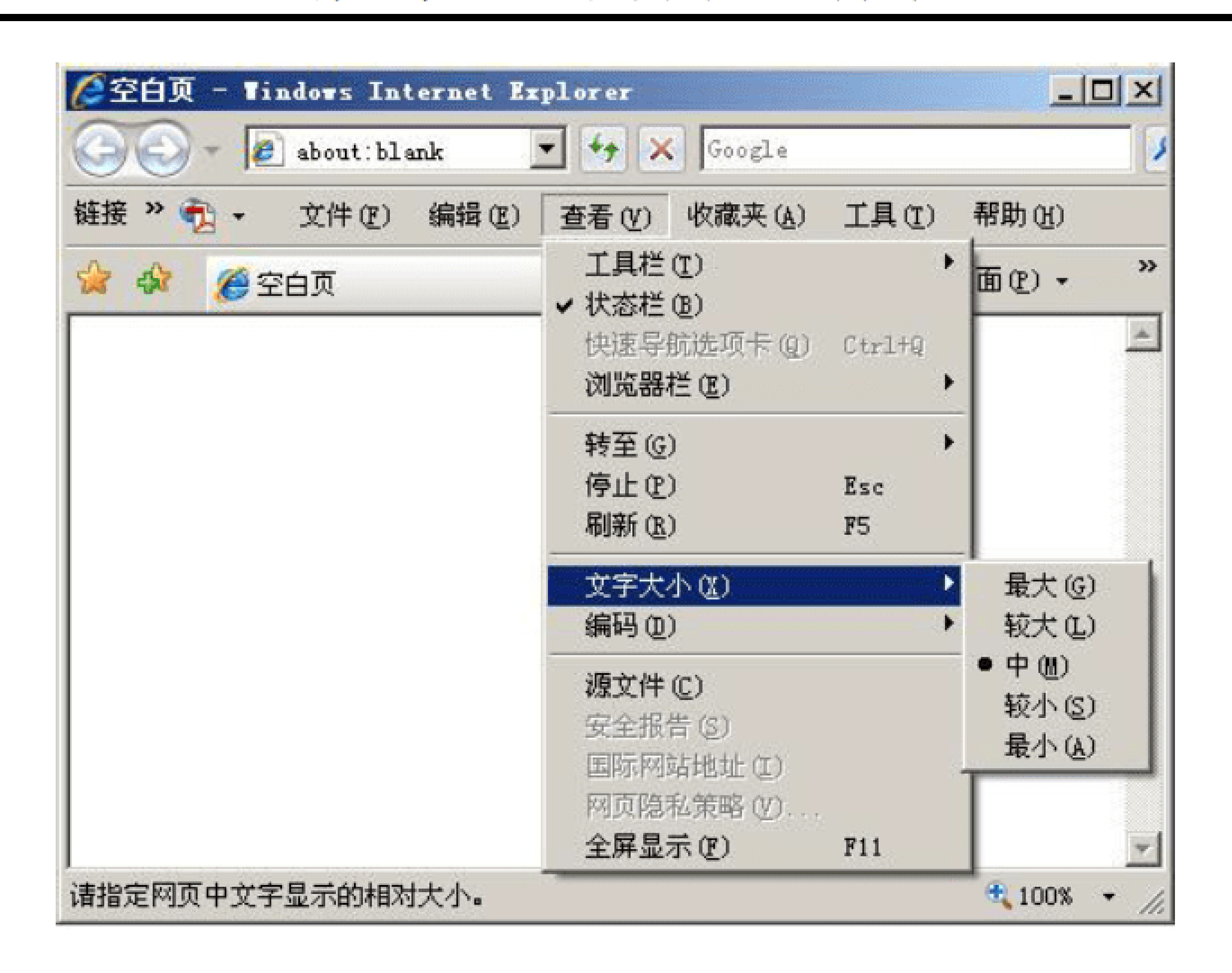


图 3-24　修改 IE 浏览器中网页显示的字体大小

【人工与自动的选择】

这一部分测试主要由人工来完成。自动进行浏览器交互测试也是可以实现的，但相比其重要性而言，实现的成本有一点不值得。这是因为正在使用的浏览器有很多种，针对每一种都需要开发自动测试平台，代码和维护的工作量较大，稳定性也不容易做的很好。另外，浏览器也是不断变化的，很可能有新的主流浏览器涌现。而且，对于 Web 应用来说，一般而言，不同浏览器下的表现大部分情况是一致的。

3.4　其他 Web 功能测试

针对 Web 应用进行的其他功能测试包括前面没有提到的所有属于功能测试的部分，很难归为一大类。它主要包括以下几个部分：Cookie 测试、Web Service 测试和网站自定义功能的测试。

3.4.1　Cookie 测试

Cookie 通常被网站用来存储用户信息和用户操作状态，以备下次访问网站或者访问其他网页时使用。Cookie 的生成过程大致如下（假设用户的浏览器打开了 Cookie 功能，这也是默认的设置）：

（1）用户访问了某个应用了 Cookie 的网站。

（2）该网站的 Web 服务器将像用户浏览器发送关于用户输入的信息等，浏览器将这些信息以特定格式的文本文件存储在本地计算机的特定目录中。

（3）当用户再次访问该网站，或者此时访问该网站的其他网页时，网站可以根据 Cookie 中存储的信息发送不同内容的页面或者显示不同的内容。

Cookie 测试的内容就是检查 Cookies 是否能正常工作，诸如下面的几个方面：

- ❑ Cookies 是否起作用，客户端计算机硬盘中是否生成了类似文件。
- ❑ 由于 Cookie 有时间上的有效期，确认 Cookie 的时效性是否设置正确。

- 网页读取、保存、修改用户信息到 Cookie 的功能是否正确地实现。
- 网页的刷新对 Cookie 的影响是否正常。
- 若在 Cookie 中保存注册信息，确认该 Cookie 中的信息，特别是设计到密码的部分已经加密。

可以通过一些工具软件来验证当前网页设置的 Cookie 内容，比如 IE 下的 IE developer toolbar、Cookies Manager 等，也可以通过编程来实现验证 Cookie 内容。要注意的是，测试 Cookie 的代码要与网页中 Cookie 读取的代码实现方法不同，否则就失去了测试的意义。

3.4.2　Web Service 测试

所谓 Web Service，实际上就是一个网络应用程序，它向使用者暴露出一个能够通过 Web 访问进行调用的接口 API。简单地说，我们能够用在自己的网页中通过编程的方法访问 Web 来调用这个应用程序。比如当前网页上有一个在线支付的按钮，单击后自动转向银行的支付页面，用户输入信用卡信息后，付款成功转回到当前网站。银行的支付页面，就可以称为一种 Web Service。

对于 Web Service 的测试，主要工作内容就是：

测试人员模拟一些用户数据，人工或者自动地提交给 Web Service，根据不同的返回值来验证功能和容错性。

【Web Service 测试的方法】

这种测试可以利用之前讲过的等价类、边界值等测试用例编写方法来模拟出用户的数据。特别需要注意的是 Web Service 的容错处理。Web Service 一般有一些参数，可以通过改变传递到 Web Service 参数的数量以及相应的数值范围来测试它的行为，从中发现容错性的 Bug。

3.4.3　Web 功能测试的一般原则

每一个网站都可能会有自己独特的功能，比如电子相册、买卖朋友、争车位等，因此，测试工程师不能满足于前文所讲述的基本 Web 测试方法，需要对当前网站特定的功能需求进行思考，找到更多可用于测试的验证方法。在这些各不相同的功能之中，有一点是它们都具备的，那就是：

所有的功能都需要用户来使用。

因此，以用户的视角来观察、使用被测试的网页才是一个最基本的功能测试方法。尽量尝试各种各样背景用户所可能进行的所有操作，遇到不顺畅、不明白的地方，要记录下来，查询网站设计说明，和其他人员（策划、美工、开发人员、用户）多沟通，这样才能把网站测试的工作做得更好。

3.5　兼容性测试与安全测试

除了很重要的功能测试之外，Web 测试还包括性能测试（本书的主要内容，将在第 4

章开始具体讲解）、兼容性测试、安全测试等。本节将介绍后两种测试。

3.5.1　兼容性测试

兼容性测试在 Web 测试方面可以说是一个比较复杂、令人头疼的事情。这是因为在用户群体的电脑中里存在各种各样的系统，各种各样的浏览器，各种各样的设置，要使得他们都能够正常浏览网站，还真不是一件很轻松的工作。根据软件的不同，兼容性测试可以分为以下几种。

- 平台测试：考察软件与操作系统之间的兼容性。
- 浏览器测试：考察软件与各浏览器之间的兼容性。
- 显示设置测试：考察软件与屏幕显示设置之间的兼容性。
- 打印效果测试：考察软件与打印设置之间的兼容性。
- 网络线路速度测试：考察软件与网络速度之间的兼容性。

以上这几个方面，归根结底都是考察被测试软件与外部环境的兼容性。为了“举一反三”，笔者在这里先解释一下平台测试的具体含义，通过它，读者对于其他的兼容性测试，也会很快理解。

【平台测试】

在用户上网的计算机当中，安装着不同的操作系统，最常见的是 Windows，不过 Windows 也有很多版本，XP sp3、Vista sp2 等，更不要说还有 Linux、Mac OS X 甚至 Solaris、HP-UX、AIX 等这些。验证网站是否支持现有的操作系统平台，即可称为平台测试。

下面将讲解上述几种兼容性测试的要点。

3.5.2　平台测试要点

由于种种原因，网站不太可能对于所有平台的最终用户都要满足，这通常是由于如果面面俱到的话，开发、测试成本较大造成的。究竟使用哪一种操作系统，取决于网站设计说明。一般来说，Windows、Linux 和 Mac OS X 的使用者占绝大多数，保证这 3 种平台用户的浏览正常已经很不错了。各种操作系统的市场份额如图 3-25 所示，这样的数据，包括后面要提到的浏览器市场份额数据，都可以在 http://marketshare.hitslink.com/report.aspx?qprid=8 通过选择不同的分类来获得。

有人可能会问，我们测试的是网页，显示在浏览器中，应该只和浏览器相关，和操作系统有什么关系呢？实际上并非如此。举个例子来说，被测试网页上有在线播放音乐的功能，那么在不同操作系统上的表现就不一样了：

- 在 XP sp3 的系统上，系统默认带有 Windows Media Player 9，网页中可以控制 Media Player 插件进行播放。
- 在 Vista 系统上，系统默认带有 Windows Media Player 11，网页中也可以通过控制 Media Player 插件来进行播放，但是由于版本不同，会有一些差别。
- 在 Mac OS X 系统上，系统默认播放器是 QuickTime，那么网页上如何放音乐呢？需要有特别的处理。

从上面的这个例子就可以看出，虽然我们测试的是网页，平台测试还是很重要的。

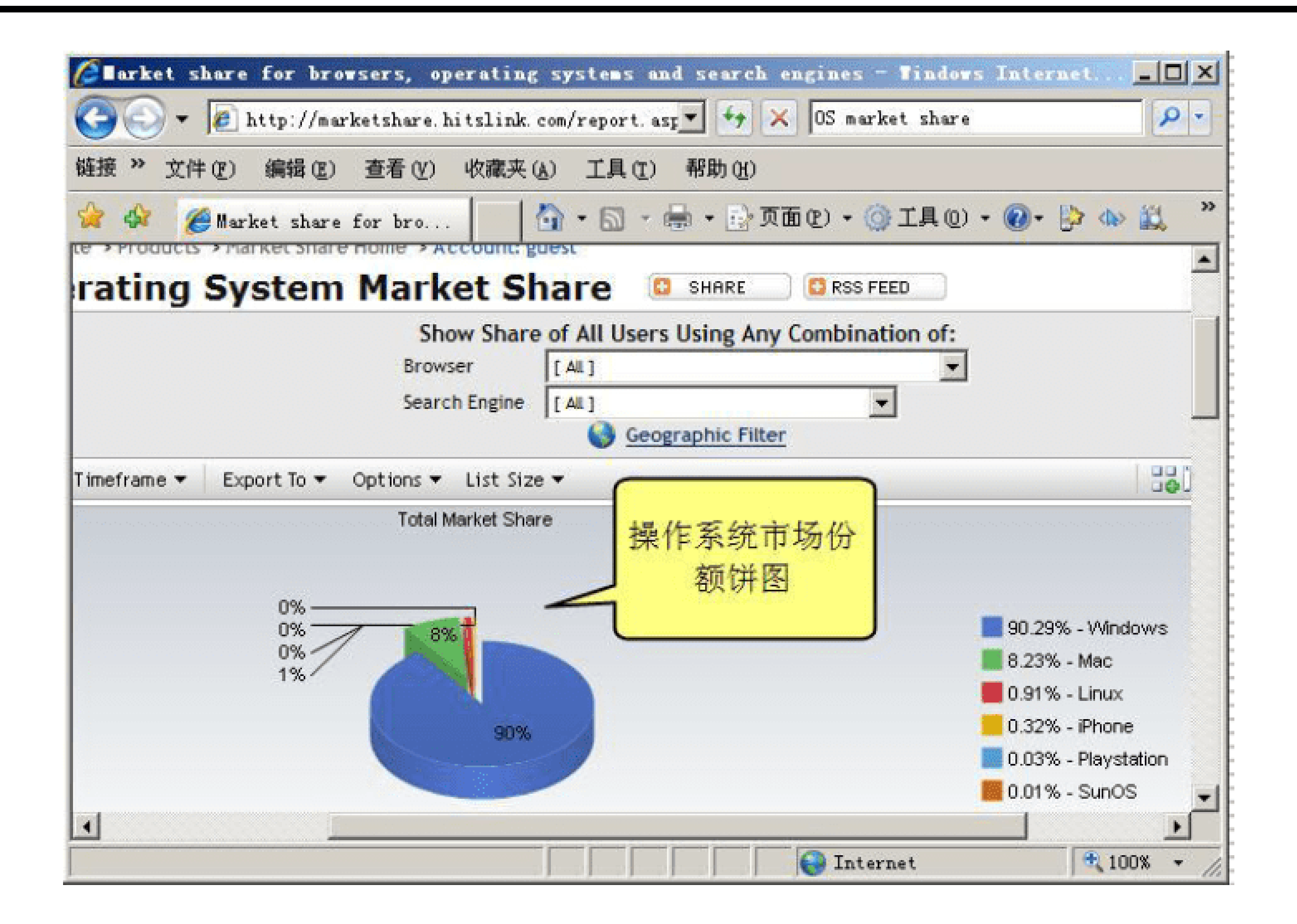


图 3-25　截至 2008 年 9 月的操作系统市场份额图

平台测试的目的就是验证网页在当前平台下能否正常工作，找到和平台相关的网页 Bug。

3.5.3　浏览器测试要点

对于网页来说，浏览器测试是非常关键的一项兼容性测试。浏览器是用户能够浏览网页最核心的软件，来自不同厂商的浏览器对 HTML 标签（比如表格间距、框架处理）、CSS 样式表（比如编写规范）、JavaScript（网页元素名称、方法名称等）、ActiveX 控件、浏览器插件 Plug-ins、安全性等诸多方面都有着程度不同的支持。代码不做调整，基本上无法完全适应所有的浏览器。

浏览器测试的目的正是为了验证网页在特定版本的某浏览器下能够正常显示。目前主流的浏览器有 IE 7、IE 6、Firefox、Safari、Opera 等，IE 8 在本书完稿的时候也已经新鲜出炉，它们各自的市场份额可以通过 3.5.2 节中查询操作系统市场份额的网址来获得，只是要选择浏览器份额的链接。

3.5.4　显示设置测试要点

显示设置主要有两部分内容。

（1）分辨率测试：网页在桌面分辨率为 640×400、600×800 或 1024×768 乃至更大的模式下是否显示正常？字体是否太小以至于无法浏览？或者是太大？文本和图片是否对齐？并找到一个最合适的分辨率作为参考和推荐。

（2）DPI 测试：网页在系统处于不同的 DPI 设置下显示是否正常？DPI 是 Dot Per Inch 的缩写，代表每英寸多少点。Windows 系统的常用 DPI 为 96 和 120。数字越大，表明字体越大。在 XP 下修改 DPI 的界面如图 3-26 所示，通过在桌面空白处右击，在弹出的快捷菜单中选择“属性”选项，在弹出窗体中单击“设置”选项卡的“高级”按钮即可发现 DPI 设置的下拉列表框（在 Vista、Windows 7 中的设置方法可能有所不同，感兴趣的读者可自

行寻找）。

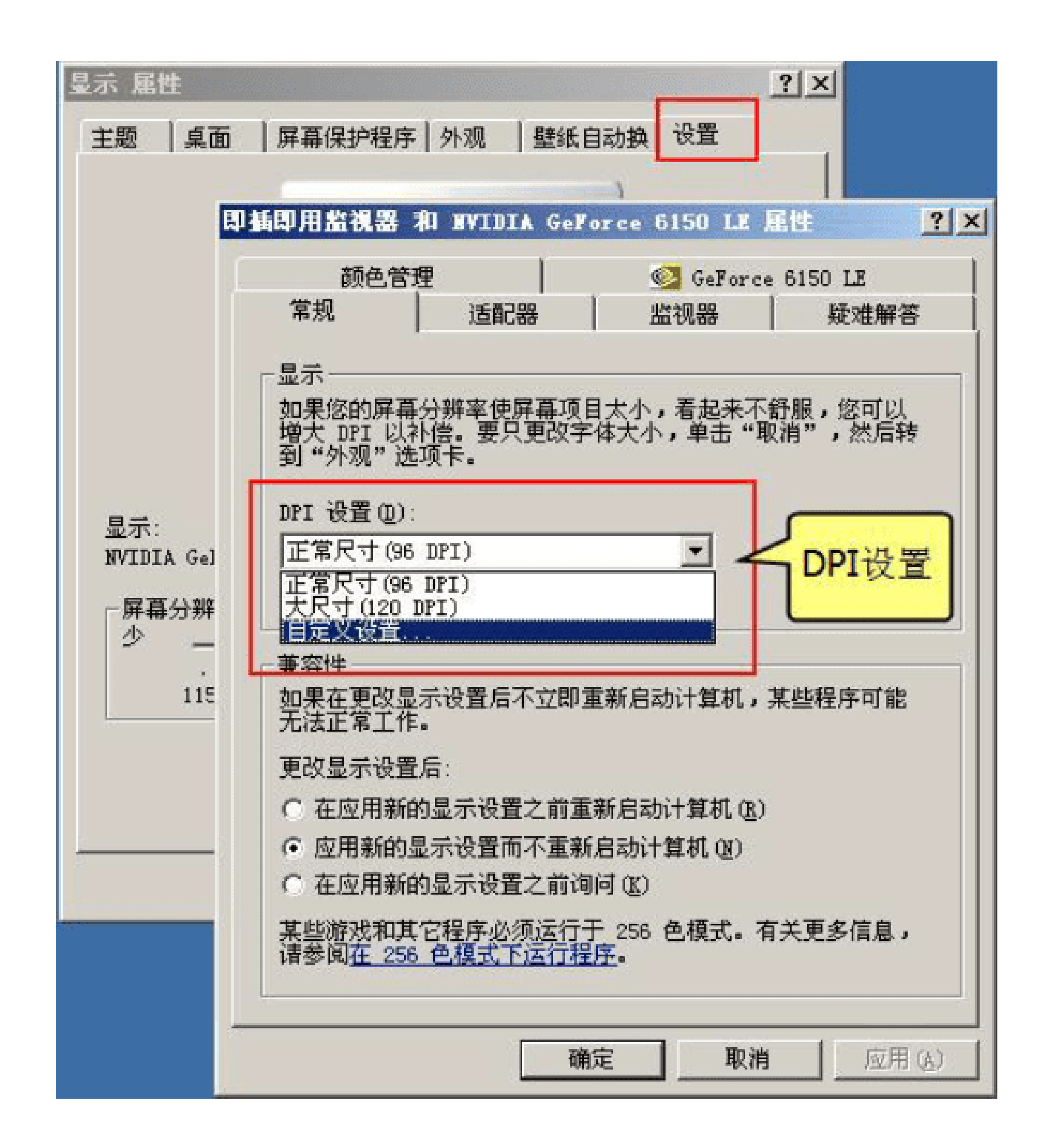


图 3-26　在 XP 下修改系统 DPI 设置

3.5.5　网络连接测试

网络连接测试主要是测试在不同连接速度下访问网页可能出现的情况。它可以说是性能测试，也可以说是兼容性测试。不管分类如何，要具备这样的意识，在网站初具雏形的时候进行一下网络连接测试，了解大致的情况，找到拖累显示速度的瓶颈，比如图片尺寸的大小、数据库连接的速度等等。

目前来讲，主流的用户上网方式有 ADSL 宽带，小区宽带和局域网接入互联网等几种方式，它们的速度分别为每秒几十 K 到几百 K 不等。网络连接测试需要模拟这些用户的不同上网条件，这种速度上的限制可以通过专门的工具软件限制带宽来实现，感兴趣的读者可以通过搜索来自行获得。

3.5.6　打印测试

对于报名表格等重要的信息，用户一般会将网页打印下来。因此网站在设计和实现的时候就要考虑到打印的问题，采用包含打印风格的 CSS 样式表（注意样式表中的 Media 类型指定为 printer）。同时，为了验证这样的样式表是否真正起作用，有必要进行人工的测试，确保网页信息能够在页面内清晰、无损失地打印出来。

3.5.7　安全测试

目前，网络所面临的安全形势是比较严峻的，各种各样针对服务器的攻击，网络黑

客……因此进行安全性测试非常必要。网络安全测试的主要目的在于：

- 验证网站本身的安全性，发现安全性问题和风险。
- 确保网站保存或者传递的用户个人信息不被盗用。

网络安全是专门的一个学科，本书这里只是简单介绍基本的入门知识。针对常见的普通攻击方法，安全测试至少要进行如下的验证：

- 确认网站目录权限设置：在测试的时候，不输入具体的网页文件名称，只输入到目录一级，验证网站的目录浏览功能是否开启。如果开启，要在网站服务器中关闭，以防止信息泄露。
- 安全套接层 SSL（Socket Security Layer）的基本验证：很多网站使用 SSL 来进行数据的加密传输。当浏览器的地址栏中协议名称变为 Https://的时候，就可以确认进入了一个支持 SSL 的站点。测试工程师需要验证进入、离开这样的页面时，无论成功还是失败时都会有较完备的提示信息。
- 用户登录信息的验证：很多网站采用登录用户才能查看或者进行更多操作的方式，以验证他们的身份。对于用户登录信息，特别是密码的保存，测试工程师要验证相关 Cookie 的设置、数据库对于密码的保存、网页 Session 超时的设置、文本框对于特殊字符的过滤、密码规则的有效性、登录失败次数的限制、登录 IP 地址的限制等功能的实现，作为网站功能测试的一部分，将测试的结果反馈给网络安全工程师等相关人员。
- 对于向 Web 服务器发送空 User-Agent、非法 Head 内容等被恶意篡改信息的情况，网站是否能正确处理。
- 还有其他的一些与安全相关的操作，由于测试工程师的网络安全知识有限，需要网络管理员、网络安全工程师等人员的多方参与。

以上这些属于基本的 Web 安全测试，如果网站在安全方面处理不好，也会带来性能问题，比如处于拒绝服务攻击（DDOS）下的情况。目前网络上有一些工具软件可供我们使用，比如微软的免费工具 WFetch，就可以用于 IIS 站点的某些安全性测试，感兴趣的读者可进一步阅读相关书籍。

由于网络安全是一个很复杂的领域，测试工程师可能无法掌握很多的专业知识，因此，在安全测试方面，测试工程师更需要做的是：按照各方（技术总监、项目经理、网络管理员、网络安全工程师、网站开发人员和测试工程师）所讨论制订出来的网站安全性说明书来进行测试，将结果反馈给各相关人员，以便及时调整测试目标，采取进一步的安全性措施，从而不断提高网站的可用性。

3.6　本 章 小 结

本章对 Web 应用和 Web 应用的测试进行了简介，这样做的目的是为了明确一个前提：**对 Web 应用进行性能测试需要首先保证它的功能是可用的。**

也就是说，在完成大部分 Web 应用的功能开发，并进行了本章所介绍过的诸多 Web 测试之后，性能测试才能提上日程，同时其测试结果也才更有意义。在本章的开始，介绍了当前 Web 应用的几大技术平台，其中较主流的是如下 3 种：

- 微软的.NET 平台；
- Sun 公司的 Java 平台；
- 开源的 PHP 平台。

另外，近年来为了提高 Web 应用与使用者的交互性，还兴起了 AJAX 技术。除了对 Web 应用的基本知识进行介绍之外，本章重点在于介绍 Web 功能测试的主要内容，其包括链接测试、表单测试、网页内容测试、用户界面测试等多种类别。

在注重功能测试的同时，兼容性测试（比如浏览器测试、平台测试），与安全性测试都是需要测试工程师在工作中特别留心的方面。对于 Web 测试，测试工程师需要建立一个从用户使用场景出发的“用户视角”方法，这样才能更好、更全面地发现问题。

有了前几章的铺垫，相信读者对于一般的 Web 应用测试已经有所了解。从第 4 章开始，将详细地讲解有关 Web 应用性能测试的知识。

第 2 篇　Web 性能测试入门

第 4 章　起点：Web 性能测试概述

从本章开始，我们将进入性能测试的世界。为了使得学习过程更加生动，还是请出我们的老朋友——小白。他经过近一个月的实习期，已经熟悉了公司内部的测试流程，掌握了一些自动测试工具软件的基本使用，但是真正的性能测试工作还没有开展。原来，公司也是第一次打算进行 Web 网站的性能测试工作。据经理说，这是因为：

公司网站是最近一年才发展起来的。之前网站的规模并不算大，注册用户并不多。测试部门的同事们非常负责，时不时地在 MSN 上询问各自的同学，让他们从全国各地尝试访问公司网站，将结果汇总到经理处，截至目前还没有出现用户抱怨速度很慢的情况。

但是，随着网站的规模越来越大，这样的土办法就不是很有科学性了，无法给管理层提供一个详细的当前网站性能状况报告，更无法预测今后网站升级、服务器端硬件升级的需求。

这正是需要小白来完成的。部门经理要求小白在最近的几个月中开展下列工作：

- 学习 Web 性能测试的基础知识，在适当的时候给测试部门的其他同事进行一次演示。
- 制定 Web 性能测试计划。
- 选择一款适合公司的 Web 性能测试软件，将报告发送给部门经理，最终希望被公司批准，成功采购。
- 利用工具软件完成一次对网站比较全面的性能测试，将报告发送给部门经理。
- 根据性能测试报告，与部门经理一起提出从软件、硬件两方面对网站性能进行优化的方案，最终交由技术总监等相关人员讨论。

这可以说是一份重任，不过也恰恰说明了公司对于性能测试的重视和对小白的信任。小白很珍惜这次挑战，作为一名性能测试工程师，他立刻开始行动起来。

不过，由于公司的网站开发已经进展了一段时间，网站初具雏形，技术总监考虑购进几台服务器用来承载网站测试版本的运行。要知道，“工欲善其事，必先利其器”，即使网站的性能测试、性能优化再完善，如果将 Web 应用部署在一台很破的奔腾电脑上也是无法达到效果的。为了有一个良好的开端，小白也参与了这几台服务器的选型会议，并有了一些收获。

在服务器选型之前，有必要对 Web 应用的性能有个基本了解。本章将从 Web 性能的定义和特点出发，附带介绍与性能关系紧密的 CPU、硬盘所起的作用。让我们和小白一起，更多地来熟悉性能测试的背景知识吧。

4.1　Web 性能的背景知识

俗话说，“巧妇难为无米之炊”。小白要学习 Web 性能测试，首先得明确以下几个

问题：

- 什么是性能？
- 什么是 Web 性能？
- 如何描述 Web 性能的好坏程度？

这些问题看似简单、基本，但却是非常关键的。在开始学习一样新事物之前，初学者最先涌上心头的往往就是这样的问题。在本节中将一一进行解答。

4.1.1　什么是 Web 性能

为了延续本书一直的风格，在笔者介绍 Web 性能之前，还是首先联想一下生活中的性能是如何进行描述的。

1．什么是性能

在日常生活中，性能这个词经常出现在汽车杂志的评测文章中，比如形容某款汽车的加速快，会说“加速性能强劲”；形容某款汽车的安全性好，会说“安全性能出众”等。因此，可以这样说，性能这个词，是通过若干考察角度（加速的能力、安全性等），对被考察物体是否符合要求的描述。而总体而言，在汽车杂志的这个例子当中，性能指标都与时间相关：

- 加速性能好，通过考察汽车从 0 起步到 96 公里/小时速度所花费的时间这一指标来说明。
- 安全性能好，可以通过汽车刹车后停车的时间（或刹车距离）来考察（当然还有其他的指标）。

【IT 领域内的性能】

在 IT 领域，由于软件、硬件并不是汽车，性能实际考察的范围会有所缩小。IT 行业所说的性能指的是：软件、硬件对于其提供的服务及时性、可用性要求的符合程度。

所谓符合及时性、可用性的要求，其实在 IT 领域内各网站、各杂志的评测文章中均可找到。比如打 3D 游戏画面清晰、细腻、流畅，说明 CPU 和显卡等性能不错，可用性要求就符合得好。比如系统运行速度很快，打开任何软件都像一阵风一样，说明 CPU 和硬盘等性能不错，即是及时性要求符合得好。

而对于 Web 性能来说，由于 Web 有自身的特点，给 IT 性能的含义带来更多的内容。

2．什么是Web性能

由于 Web 应用是基于交互式的（请读者回忆一下第 3 章所介绍的浏览网站过程中请求—反馈两个动作），使得 Web 性能更偏重于对及时性要求的满足（即服务要及时地给予用户的请求以反馈）。当然，Web 性能中的可用性也是不可忽略的，它主要描述了 Web 应用在正常的情况下最多能够同时给予多少用户以服务，以及服务能正常持续多长时间。

3．如何描述Web性能的好坏

类似汽车业界的若干标准，IT 领域也需要制定一些具体的指标，以描述 Web 性能的好坏。具体地说，Web 性能的及时性和可用性由下面两个基本的指标来衡量：

- 响应时间（Response Time）来衡量及时性。所谓响应时间，即 Web 应用对于用户请求做出响应，再发送反馈直至用户接收完毕所需要的时间。
- 最大并发用户数（Concurrent User）来衡量可用性。最大并发用户数量是 Web 应用在不出现系统崩溃的情况下，所能同时提供服务的最大用户数量。

除此之外，还有众多相关指标，但以上两点指标就可以说明大部分性能问题。它们与及时性、可用性要求的符合程度，分别可以用性能测试和压力测试来获得。不过，在很多情况下业内把压力测试也归于性能测试的范围之内。本书所指的性能测试也采用了这样的广义范围，这一点需要读者注意。

4.1.2　Web 性能的影响

上一节介绍了 Web 性能的两种指标，那它们的好坏对于我们有什么影响呢？这个问题其实有多个回答，因为对于不同的人群，影响也不同。

简单说来，与当前 Web 应用相关的人都会受到不同程度的影响，他们分别是：

- Web 应用的使用者，绝大情况下就是网站的最终用户。
- Web 应用的管理者，也就是网站的管理员。
- Web 应用的测试者，也就是小白这样的测试工程师。
- Web 应用的开发者，即网站的开发人员。

1. 用户感觉到的响应时间

最终用户，是英文名称 End User 的直译。从其字面“最终的使用者”就可以明白，网站最终是要交付给用户来使用的。对最终用户来说，Web 性能好坏的判定标准就是网站对用户操作的响应时间。这段时间会是以下几种甚至它们的叠加：

- 用户输入网站网址、按下 Enter 键之后，直到网站首页基本打开的时间间隔，为了描述方便，简称为“打开网站时间”。
- 用户单击网页上的各类“提交”（Submit）按钮，直到服务器返回完全结果的时间间隔。同样，本书简称为“操作完成时间”。

对于用户来说，以上的响应时间是感觉上的，不会有一个具体、统一的度量标准；而且，这样的响应时间有时候并不完全与网站相关，比如下面的情况：

- 对于“打开网站时间”，由于在浏览器中输入网址后需要查询网络上的 DNS 记录，之后才是真正连接到网站，这个过程需要花费一点时间，但也被包含在用户感觉到的响应时间内了。有时候 DNS 服务器会出现问题，导致网站响应时间非常长，但和网站的 Web 性能无关。
- 对于“操作完成的时间”，实践证明，如果网页上已经出现了操作结果，哪怕只是小部分，用户的满意度就会上升，其感受到的响应时间会比显示完全结果的“操作完成时间”短。这也是很多网站在返回大量结果时，用进度条或者提示提醒用户等待、用分页拆分显示完全结果的原因。如图 4-1 所示为某中型网站搜索页面在用户输入关键词后等待时的提示。对于用户来说，网页显示部分结果、甚至是友好的内容，总比漫长的等待要强得多，会提高用户的满意度。

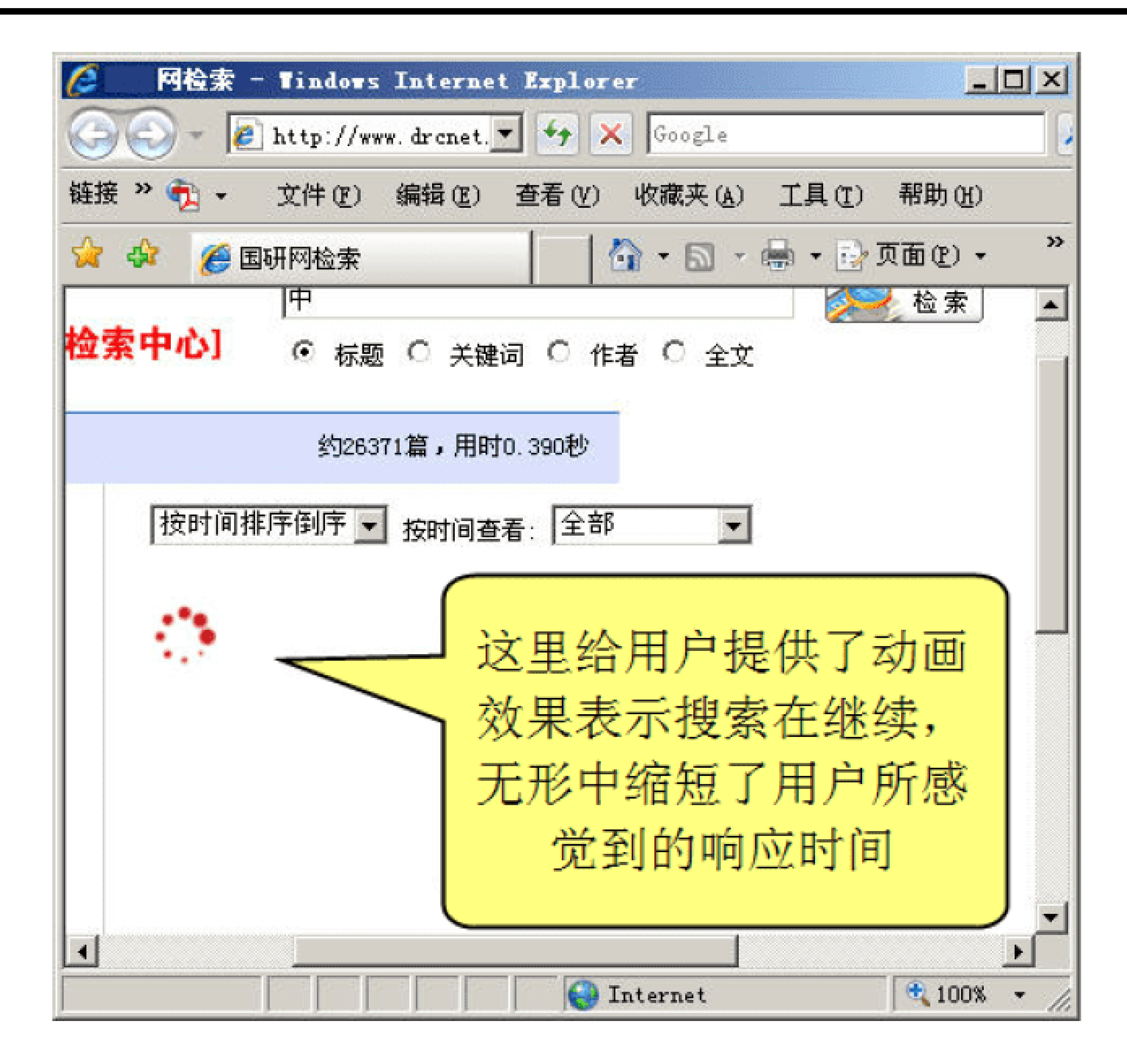


图 4-1　某网站在搜索等待时的提示

2．Web性能对用户的影响

Web 应用具备良好的性能，用户往往不会夸赞，甚至无法觉察到这种优点。但是，差的 Web 性能对用户和自身的影响却是显而易见的。

- 用户将失去继续浏览网站的耐心，造成 Web 应用用户群体、被使用功能的减少。
- 用户将对网站留下不好的印象，继而影响公司各方面（产品、服务等）的形象。

这两种影响对于网站用户来说，并没有多大的损失，因为还有其他很多类似的网站可供浏览和获取信息；而对于网站来说，就是比较大的损失了。大部分网站是要争夺最终用户的，并且尽可能地希望用户在网站上停留时间多一点。从上面两点来看，差的 Web 性能将极大地损害用户满意度，降低用户忠诚度。

3．Web性能对网站管理员的影响

经过测试工程师测试合格之后，Web 应用具备了上线交付使用的条件。不同于网站用户对网站性能的置身事外，Web 性能变差对于网站管理员的影响是非常大的。这是因为：网站管理员除了关注用户体验之外，更要关心网站各服务器的状态信息。如果出现了 Web 性能变差的情况，网站管理员需要采取如下的行动：

- 进一步监控系统状态（网络环境等）、服务器资源状况（CPU，硬盘，内存等）、应用服务器和数据库的资源状况，找到 Web 性能瓶颈产生的原因。
- 通过系统硬件、Web 应用所依赖的软件（比如数据库系统、网站应用服务器等）的优化设置在短期内尽量提高或者保证 Web 性能。
- 制定相应的预防措施、升级计划，预防长期内的 Web 性能变差出现。

这几条措施都是为了保证网站的稳定性和扩展性，也是网站管理员的工作职责。在其中，根据实际情况，也需要服务器设备技术支持人员、Web 性能测试工程师、Web 应用开发人员等参与。

相对于下面马上要提到的性能测试工程师，网络管理员更关注于网站上线后，系统硬

件资源的表现。

4．Web性能对性能测试工程师的影响

性能测试工程师更关注的是软件代码对于系统硬件资源的作用。

如果网站尚未正式上线，那么其测试版本的 Web 性能变差对于性能测试工程师未必不是一件好事。这不仅意味着性能测试的工作非常重要，而且小白的工作也会遇到挑战。正如本章开头部门经理布置给小白的工作任务，测试工程师将对网站的测试版本进行性能测试并提出改进意见。注意，这样的测试采用的都是模拟的数据，获得的结果将提供给网站开发人员、网络管理员做改进参考。

如果网站已经正式上线，性能测试工程师也可以和网络管理员一起，分析 Web 性能的瓶颈所在，提出优化和改进建议；另外，还要分析之前模拟数据的测试结果和当前真实数据结果之间的差异，为下一次性能测试积累经验。

5．Web性能对开发工程师的影响

Web 性能变差对于开发工程师的影响相对于前几类人群来说是最大的。这是因为，在同样的硬件资源配置下，Web 应用程序中对资源的分配好坏与否产生的结果截然不同，而只有开发工程师能够对其进行修改。

开发工程师不仅要获知用户的响应时间、网络管理员提供的系统资源信息、分析测试工程师的测试报告，还要钻研并解决以下几个问题：

- 宏观方面，如何通过调整网站的技术平台、具体设计、相关的代码实现或者数据库等依赖软件的设置，提高总体 Web 性能。
- 微观方面，发现、解决网站程序中分配资源代码的 Bug，比如内存使用是否合理、多线程是否必要、申请对象是否及时释放、资源竞争是否会出现死锁等问题，使得网站并发用户数量增加，稳定性提高。

【小总结：荷包蛋图】

从以上的叙述可以看出，用户、Web 性能测试工程师、网络管理员、Web 应用开发工程师对于性能的关注是从黑盒到白盒、从硬件系统到软件代码、从浅入深的，如图 4-2 中的“荷包蛋”所示。因而，他们面对 Web 性能变差后所需要做的工作，即对其的影响也是依次从小到大的。

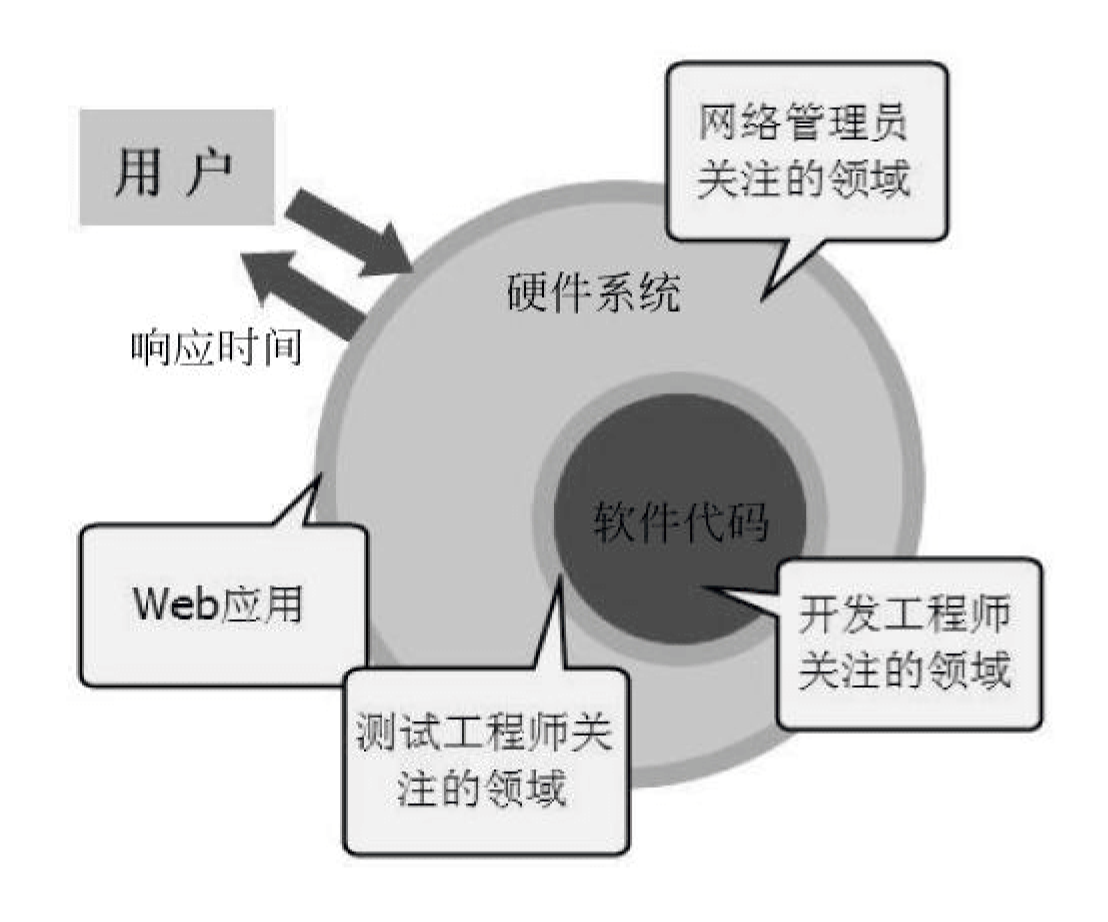


图 4-2　不同人群对于 Web 性能的关注图

在前面的叙述中，我们已经提到了网站上线、性能测试工程师发送测试报告、网站运行时网络管理员收集硬件资源信息等包含时间概念的词语，那么就有必要对业内网站开发的大致过程做个简单的介绍，这将是下一小节的内容。

4.1.3　Web 性能测试在网站开发中的位置

与第 1 章所说的软件项目开发流程类似也有所区别，目前国内网站开发的大致过程如下：

- 网站准备阶段：网站策划，栏目、功能设置等。
- 网站设计阶段：美工、用户界面工程师设计出网站效果图并最终确定页面外观。
- 网站开发、测试阶段：美工按照既定页面外观完成所有页面、图片等设计；网页制作人员编写 HTML 页面和部分客户端脚本；网站开发工程师编写代码；测试工程师测试网站功能，并根据需要进行安全测试、性能测试、多语言测试。
- 网站初始内容上传：网站编辑上传内容。
- 网站上线：网站技术人员、网站管理员部署网站代码，系统上线。
- 网站维护：网站管理员维护网站正常运行；网站编辑持续上传最新内容；美工和网页制作人员更新页面、图片；网站开发、测试工程师编写和测试新功能代码。

表 4-1 描述了这些角色所使用的主要技术和工作成果。

表 4-1　网站开发流程中的各个技术角色

技 术 角 色	主 要 技 术	工作开始的前提条件	工作产生的结果
页面设计	平面设计以及相应工具 Photoshop、Fireworks、Flash 等	客户的需求 上级的要求	页面效果图
页面制作	HTML、CSS、DOM 和 JavsScript 知识	页面效果图基本完成	HTML 页面，或者叫做静态页面
Web 开发	服务器端程序开发知识，比如 C#，Java，数据库知识	网站功能设计说明基本完成	可以部署到服务器上的动态程序，一般以 aspx，jsp 等为文件名后缀
Web 测试	测试相关知识，与当前使用开发平台相关的知识	网站功能设计说明基本完成、已经有部分实现代码	测试工具、测试报告、Bug 和测试用例
网站管理（包括数据库管理员）	网络相关知识，操作系统与数据库管理的相关知识	网站上线	网站维护报告、数据库备份文件等

可以看到，在前面的开发过程和表 4-1 中，测试阶段处于中间阶段，而性能测试在 Web 测试中一般处于后期，因此性能测试在整个开发流程中处于：

- Web 开发接近完成。
- 网站上线前。

这是因为两个原因：

- 如果 Web 开发尚处于开始阶段，很多功能尚未实现，代码还处于剧烈变动的时期，每天都会发生很大变化，此时测试性能会造成很多的重复劳动。
- 网站上线前的一段时间，由于代码的变动已经很少，结构趋于稳定，同时对于网

站用户的定位也更清晰，可以做多次性能测试，来模拟真实的场景，并据此改进，直到达到要求或者到达预定发布日期。

4.1.4　Web 性能测试的目的

基于以上的介绍，我们可以总结出对 Web 应用进行性能测试有如下几个目的：

- 发现系统的代码缺陷：这是性能测试首先可能发现的问题。如果性能测试结果（主要指 Web 应用的响应时间、服务器资源占用情况）与预期的相差较远，需要考虑这种性能变差是否由于 Web 应用的程序代码中的 Bug。
- 发现系统的工作能力：在可能的程序代码 Bug 基本解决，不再出现新的类似 Bug 之后，进行性能测试可以验证被测试的 Web 应用在预设环境下具备何种程度的工作能力。
- 发现系统性能优化的关键点：在发现系统的工作能力后，根据性能测试结果，寻找其中可以进一步改进的部分，通过这些部分的比较与模拟测试，进一步给出系统性能优化建议。

以上 3 个部分是递进的，一般随着项目进度的进展而先后实施。它们也可能循环多次进行，即针对每一次的最终结果再从头开始，找出下一步的性能优化关键点，这种方法在本书之前介绍过，称为迭代（Iteration）。当然，具体迭代次数在实际工作中，要依据时间、要求以及人力物力资源的充裕度而定。

4.2　影响 Web 性能的重要硬件 I：CPU

在了解 Web 性能的简单背景知识之后，小白第 2 天便兴冲冲地旁听了网站服务器的技术选型工作会议。所谓技术选型，实际就是综合考虑网站的技术和特点，选择用何种型号的服务器承载 Web 应用。除了很重要的预算、成本因素之外，小白在这次会议上还了解了一些 Web 应用性能与服务器硬件之间的基本知识：通过提高两个重要硬件，CPU 和硬盘的性能来提升 Web 应用性能的方法。本节就将介绍他获得的这些经验。

4.2.1　中央处理器（CPU）简介

中央处理器，英文名称为 Central Processing Unit，直译就是中央处理单元。中央处理器是计算机的核心，其重要性好比大脑或者心脏对于人一样。在功能上，处理器的作用更接近于大脑，负责处理、计算电脑内部传递的所有数据。CPU 的类型决定了可以使用的操作系统以及相应的软件。比如：

装有 64 位 CPU 的电脑既可以安装 32 位的操作系统（比如 Windows 2003 标准版），也可以安装 64 位的操作系统（比如 Windows 2008 64 位版本）。反之则不可以：装有 32 位 CPU 的电脑无法安装 64 位操作系统。

CPU 主要由运算器、控制器、寄存器组和内部总线等构成。

目前市场上主流的 CPU 主要由 Intel 公司和 AMD 公司生产，品牌分别有 Core 2 Duo、

Celeron、Pentium、Athlon 等多种，另各自有内部的型号代表不同的性能参数组合。Intel 的某一种 CPU 外观如图 4-3 所示。所有的 CPU 外观与此大同小异。

现在我们的小白遇到了一个问题，他现在有 3 台服务器，CPU 的指标不同，如何找出在哪一台上安装网站代码会获得更好的性能呢？方法是利用工具软件。

在这里，推荐一款在 Windows 平台下很有名、免费又好用的工具软件——CPU-Z。它不仅能获取 CPU 的信息，还能获取一些主板与内存的概要信息。



图 4-3　一种 Intel CPU 的外观图

4.2.2　CPU-Z 简介

CPU-Z 是一款免费的获取系统信息的工具软件，可以从下面的官方网站上下载：

http://www.cpuid.com/cpuz.php。

其汉化版本也可以在国内各大软件下载网站，比如太平洋下载中心、天空软件站等找到。截至 2008 年 9 月，CPU-Z 的最新版本为 1.47 版。CPU-Z 可以获取的系统信息如表 4-2 所示。

表 4-2　CPU-Z软件可以获取的系统信息

硬 件 类 别	可获取的信息
CPU	名称、序列号、CPU 代号 核心电压 内部、外部时钟频率与倍数关系 支持的指令集 缓存（Cache）信息等
主板	制造商、型号、版本 BIOS 型号与日期 芯片组、感应器信息 显卡型号
内存	频率和时间周期 制造商、序列号等信息
系统	Windows 版本 DirectX 版本

4.2.3　CPU-Z 的使用方法

单击网页上的下载 CPU-Z 链接，会下载一个 zip 文件，解压缩后，得到一个文件夹，其名称包含当前的版本。比如最新版本是 1.47，那么解压缩后生成的文件夹名称即为 cpuz_147，如图 4-4 所示。

可以发现，文件夹中非常简单，只有一个主程序、一个配置文件和一个简单的使用说明文件。在运行 CPU-Z 时，它不会在磁盘上写文件，也不会读写注册表，除了主程序之外，不依赖任何额外的系统文件、特别功能的支持，也可以称得上是一款“绿色”软件了。CPU-Z 正是由于它的使用简单、信息全面赢得了很好的口碑。



图 4-4　CPU-Z 下载并解压缩后生成的文件夹内容

单击 CPU-Z.exe，经过几秒钟的收集信息等待时间，弹出界面如图 4-5 所示。可以看到，在图片的上方，用 Processor 围住的方框内，列出了当前电脑的 CPU 信息，如表 4-3 所示。

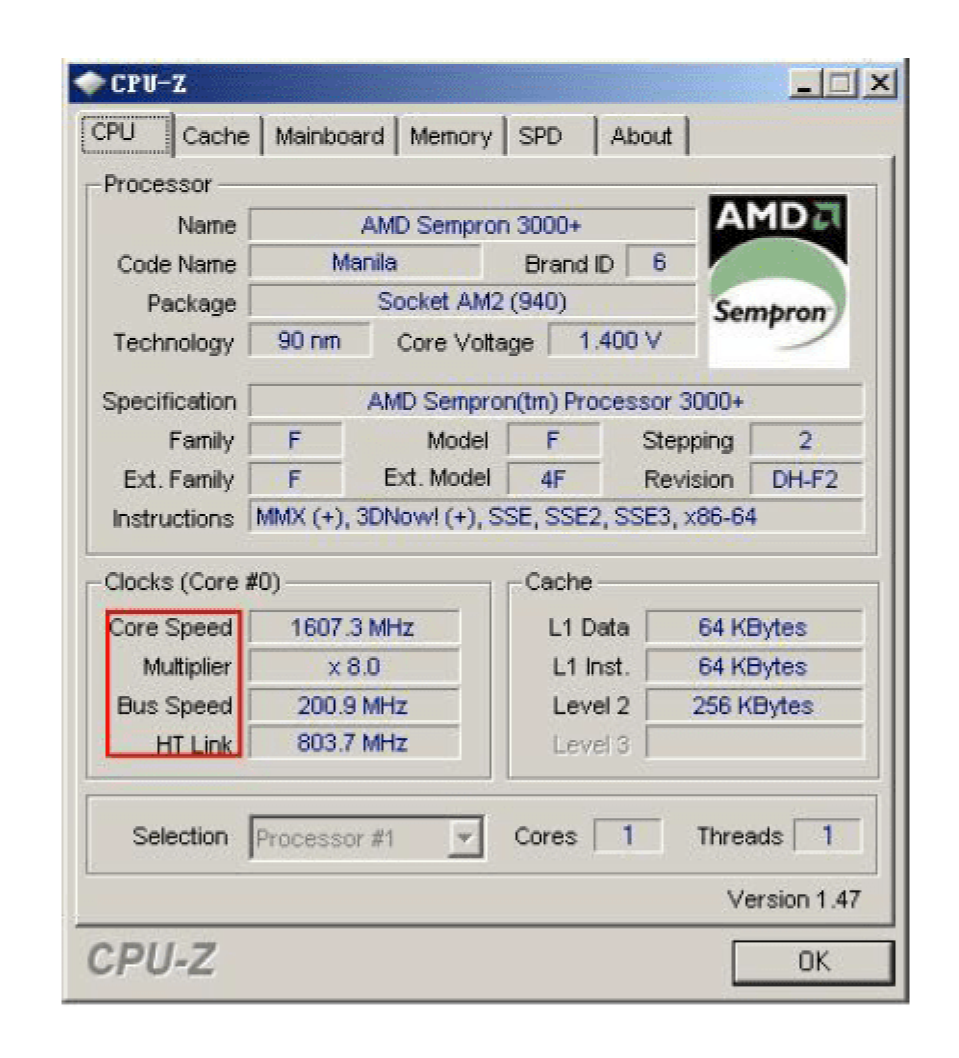


图 4-5　某台电脑上运行 CPU-Z 的结果

表 4-3　图 4-12 所显示的CPU信息

指 标 名 称	值 与 简 介
Name/CPU 名称	AMD Sempron 3000+ AMD 公司的型号
Code Name/CPU 代码	Manila 产品代号：马尼拉，菲律宾首都
Package/CPU 插座标准	Socket AM2 940 针 CPU 背面针脚的数量
Technology/CPU 晶片技术	90nm 越小越现代，最新一般是 45nm
Core Voltage/核心电压	1.424V 这个值随系统运行会不断变化，越低 CPU 耗能越低

剩下的一些信息稍微深入了一点，在此不列举了。在图 4-5 中，我们注意到标出了一个红色的方框，这其中的几个指标就是下面即将介绍的，它们的优劣将对 Web 性能产生显著的影响。

在 CPU-Z 的其他标签（Cache 主板缓存、Mainboard 主板信息、Memory 内存信息等）下也可以找到对应的信息，但是一般来说，它们对于速度的影响并不如 CPU 的这些数据来得大。

【核心速度：Core Speed】

所谓核心速度就是指 CPU 的速度，以 MHz 为单位。我们只需要知道数字越大，说明 CPU 速度越快，在其他指标一致的情况下，电脑也越快。

有些时候，CPU-Z 显示的核心速度和说明书有所差别，这是很正常的，因为现在很多 CPU 都有在系统工作不繁忙的时候降低速度运行的功能，这样做是为了减低功耗、延长寿命。因此，在运行 CPU-Z 的时候，系统最好还是处于一定的负荷当中。

【倍频：Multiplier】

简单地说，倍频指的是 CPU 核心速度与连接 CPU 和其他芯片之间数据线（地址/数据总线、或者叫做前端总线 FSB）速度的比率。这二者的速度都是以频率为指标，即单位时间内执行命令的次数。如果倍频等于 8，也就是说单位时间内，CPU 执行了 8 个命令，而前端总线只执行了一个命令。在图 4-5 中也可以发现，核心速度的值是总线速度的 8 倍。

200.9MHz×8 = 1607.3MHz

【总线速度：Bus Speed】

在前面的倍频中，我们已经提到了总线速度，它实际指的是前端总线（和 CPU 相连的，英文名称为 Front Side Bus）的速度。前端总线负责在 CPU 和北桥芯片之间传递数据。在图 4-5 中 Bus Speed 下方还有一个指标：HT Link，这是 AMD 公司生产的某些芯片所独有的一项技术，类似 FSB 所起的作用，我们不必过多关注。

上面提到的这几项指标对于计算机速度的影响，通过一个典型 PC 机主板内部的概要结构图就可以很清楚，如图 4-6 所示。

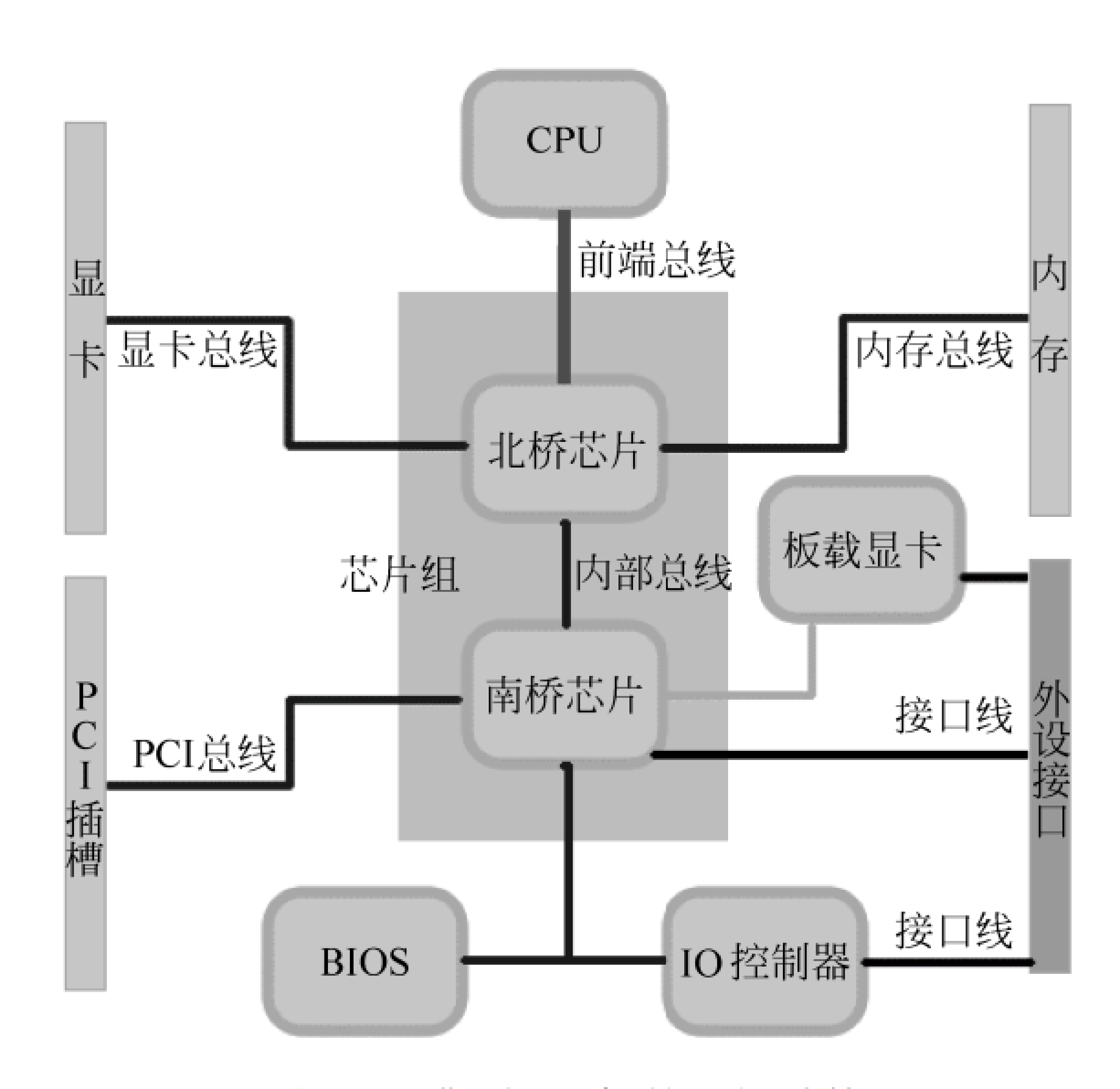


图 4-6　典型 PC 机的主板结构

在图 4-6 中，前端总线处于 CPU 和北桥芯片之间的位置，CPU 计算速度再快，数据总要经过前端总线运送出去，如果这条马路出现了拥堵，可想而知电脑的运行速度也不会很快。

【多核的问题】

目前很多中央处理器都是多核的，比如双核、4 核，这样它同时处理数据的能力就大为增强，类似寺院里的“千手观音”：手多代表能处理人间疾苦的能力强。针对 Web 应用来说，同一时刻能满足更多的用户浏览是很重要的，因此，多核的 CPU 对于改善 Web 应用性能很有好处。而对于有些需要大规模单线程计算的应用来说，比如生成源代码的 Build 系统，多核反倒有点浪费，只需要关注核心速度和总线速度即可。

在 CPU-Z 的其他标签中还有一些有关 CPU 的信息，有兴趣的读者可以在工作中做进一步的了解。

【适合 Web 应用的 CPU 小总结】

因此，在选择 Web 应用的服务器 CPU 时，要尽量选择核心速度和 FSB、HT Link 速度数值都更大、多核的中央处理器。好在这个标准很好满足：一般来说越新的处理器，这两个数值就越大，核心也越多。

同样地，从图 4-6 中也可以看出，内存总线和内存速度也类似前端总线与 CPU 核心速度这一对的情况，因此在这里就不赘述了。另外，在购买服务器的时候，生产厂家根据型号一般会搭配好内存，而 CPU 则需要自己选择。

在 CPU 讲解的最后，以笔者曾经做过的一次性能测试为例，说明硬件的选择对于性能的巨大影响。笔者在两台服务器上分别测试了某应用所花费的执行时间，结果如表 4-4 所示。

表 4-4　某次性能测试的结果

服务器型号	服务器价格	性能测试结果
服务器 A FSB：1333MHz 核心速度：3.0G	55000 元	46 分钟
服务器 B FSB：1033MHz 核心速度：3.2G	104000 元	83 分钟

由此可见，并不是越贵的服务器速度就越快，而应该是从要完成任务的类型出发来选择合适的硬件。这一规则是非常重要的，违背它可能会产生事倍功半的效果。

4.3　影响 Web 性能的重要硬件 II：硬盘

硬盘，在一些文章中也被称简写为 HDD（Hard Disk Drive）、FDD（Fixed Disk Drive），是计算机中存储数据的关键设备。Web 应用的源代码、网站和用户信息的数据库都会存放在这里，因此，硬盘对于 Web 性能也起到了比较重要的作用。在选择什么样的服务器来承载 Web 应用以获得更好的性能过程中，硬盘有几个指标需要小白来考虑。

- 硬盘的类型：即硬盘采用什么方式存储数据、传输数据。
- 硬盘的转速：硬盘中碟片的转速。
- 硬盘的缓存：其大小代表硬盘为了读取速度更快而设置的缓存容量。

4.3.1　硬盘的类型

目前，主流的硬盘有 4 种类型，如表 4-5 所示。

表 4-5　主流硬盘的 4 种类型

类　　型	全　　称	特　　点
SCSI	Small Computer System Interface	60 针/80 针接口，速度 160M/s 至 640M/s 不等
SAS	Serial Attached SCSI	串行版本的 SCSI，减少了线缆干扰，速度更快
ATA	Advanced Technology Attachment	个人电脑常见硬盘接口，133M/s 左右
SATA	Serial ATA	个人电脑常见硬盘接口，150M/s 左右

表 4-5 中列出的硬盘类型内部还会细分为不同的子标准，因为电子产品升级换代是很迅速的，不过这些子标准的速度大致在一个数量级上。对于承载 Web 应用的服务器来说，硬盘类型一般是 SCSI 和 SAS 两种，建议选择更新的 SAS 硬盘。

4.3.2　硬盘的转速

硬盘读写数据的原理很复杂。简单说来，硬盘是由一个或者多个摞起来的“盘子”组成的。“盘子”类似 CD 光盘，每圈均记录有数据。磁头悬浮在每个“盘子”的表面。系统通电后，“盘子”转动，磁头通过马达带动在“盘子”表面上方来回移动，从而完成对其上存储的信息进行读写的操作。硬盘简单的结构图如图 4-7 所示。

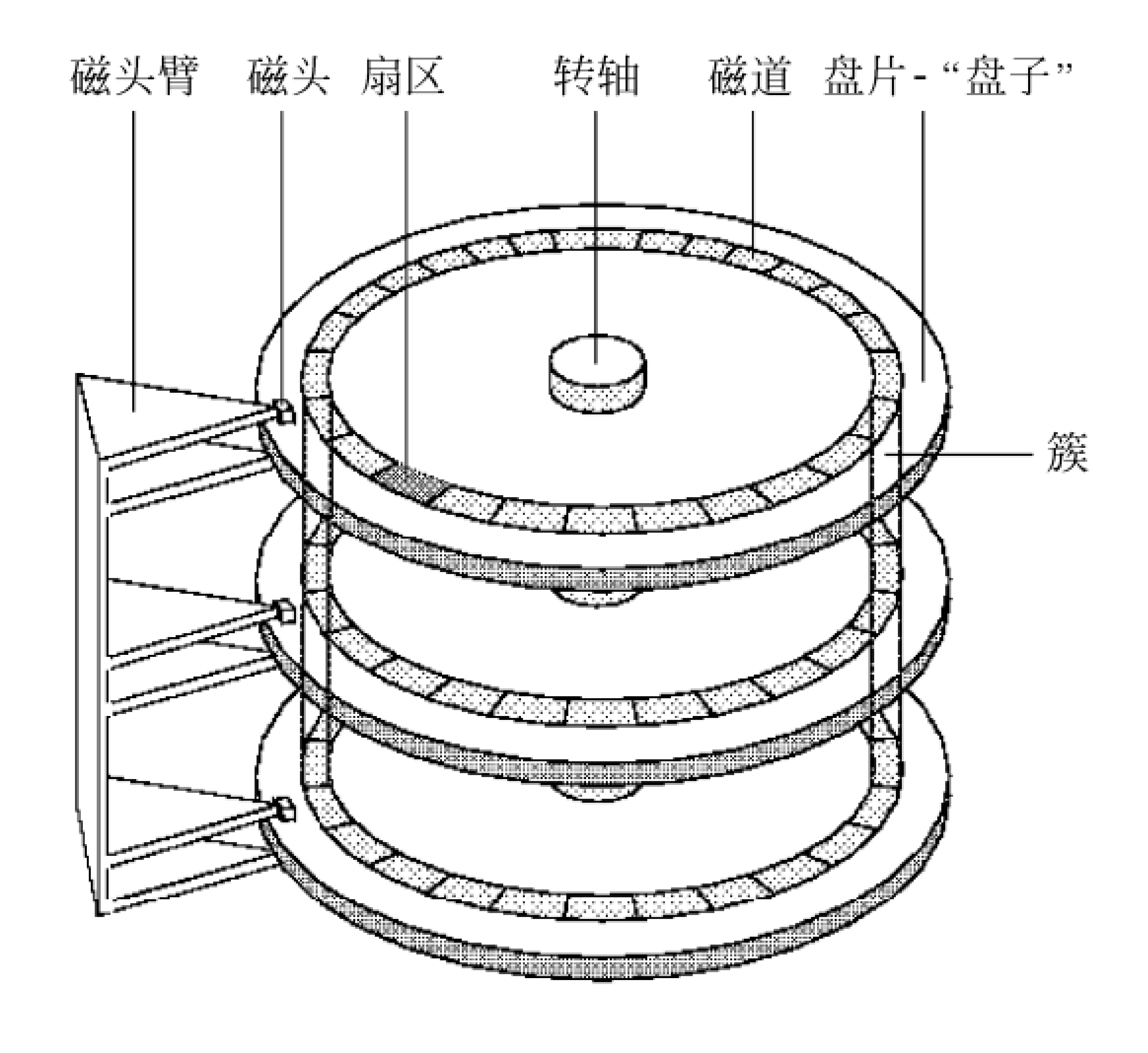


图 4-7　硬盘的简单结构图

在图 4-7 中，“盘子”上每圈磁道包含多个扇区，扇区中存储着数据。磁头在扇区之间穿梭读写。因此可以想象，如果“盘子”的转速快，自然磁头从一个扇区到另外一个扇区所花的时间（一般来说）是会变短的。每次读写扇区的时间都变短一点，积累下来就很可观了。从这里可以简单地推断，硬盘的转速越快，读写数据的速度越快，因此对于用户查找网站中的数据及进行的操作响应的就越快，Web 性能越好。当然，硬盘转速不是可以无限快的，它的很多零部件毕竟是机械构造的。目前，主流的硬盘一般有 5400 转/分钟（笔

记本硬盘）、7200 转/分和 10000 转/分等多种规格。在选择硬盘方面，本着提高 Web 性能的原则，转速越高越好。

4.3.3　硬盘缓存

硬盘的缓存可以分为硬件和软件两种。下面我们先来学习硬件上的缓存——硬盘缓存。

硬盘缓存处于硬盘背板上，是硬盘上的一个存储芯片，一般容量为几兆。缓存好比我们家里的衣柜，当前季节的衣服一般放在衣柜的外面，便于经常穿着；过季的衣服和被子等一般放置于衣柜的最里边甚至床下，这是因为它们在当前并不是需要的东西。硬盘缓存（当然还有其他形式的缓存，比如 CPU 缓存等）和生活中的这种情况类似，因为硬盘内外数据传递速度显著不同，这是为了数据传输的方便而设计的。

硬盘缓存主要有以下 3 大作用：

1．预先读取

当硬盘接收 CPU 传过来的指令开始读取数据时，硬盘控制芯片会控制磁头将下一次要读的数据预先送到硬盘缓存中。由于缓存是一个芯片，电信号的运行速度远远高于磁头读写的机械速度，因此，能够明显改善性能。

2．对磁头写入数据的操作进行缓存

当硬盘接到写入数据指令后，如果立刻将数据写入硬盘，就会导致磁头运动过于频繁，影响硬盘寿命。而如果先将数据存储在硬盘缓存里，等到缓存数据达到一定要求的时候再写入硬盘，就可以减少磁头来回运动的消耗，有利于提高速度和性能。

这个也同样可以用生活中的例子来说明。小白在家洗了一堆衣服，晾干后，是每次收一件就把它放到衣柜里快，还是都收下来一次性地放在衣柜里快呢？相信大多数情况下后者会快些。

3．临时存储最近访问过的数据

对于 Web 应用来说，经常需要频繁访问某个数据，比如用户名、登录状态、记录的 IP 地址、首页的图片等，那么有了缓存之后，就可以直接从硬盘缓存中调用这些数据，不用劳磁头机械手的“大驾”了。这个办法同样减少了磁头的机械运动（耗费时间，磨损硬盘），提高了性能。

不过值得注意的是，一旦电脑关机，缓存由于是芯片，里边保存的数据会被清空。硬盘缓存越大，自然能减少的磁头运动就越多，对性能的提升就越明显。

4.3.4　操作系统中的硬盘写入缓存

操作系统通过软件方式也提供了硬盘写入缓存的功能，其具体原理不是本书所探讨的内容，这里只介绍一下如何开启这样的功能以获得最大可能的性能。

在 Windows XP（Windows Vista、2003 中也有，菜单略有不同，读者可自行发现）中，单击“开始”按钮，按照如图 4-8 所示的操作顺序打开计算机管理界面。

（1）选择“管理”选项后，出现“计算机管理”对话框，如图 4-9 所示。



图 4-8　打开计算机管理



图 4-9　选择磁盘

（2）选择好磁盘分区后，右击分区在弹出的快捷菜单中选择“属性”选项，在弹出的该磁盘分区属性窗体中选择“硬件”标签，进入“硬件”选项卡。在选择“所有磁盘驱动器”列表框中选择某个硬盘，再单击对话框下部的“属性”按钮，即可对磁盘属性进行设置，如图 4-10 所示。

（3）在图 4-10 中单击右下方的“属性”按钮后，会弹出所选择物理磁盘的属性对话框，如图 4-11 所示。在“写入缓存和安全删除”选项区域中，确认“为提高性能而优化”被选中，并将下面的一个或几个复选框（视系统、硬盘不同而不同）都选中，单击“确定”按钮使设置生效。



图 4-10　打开磁盘属性

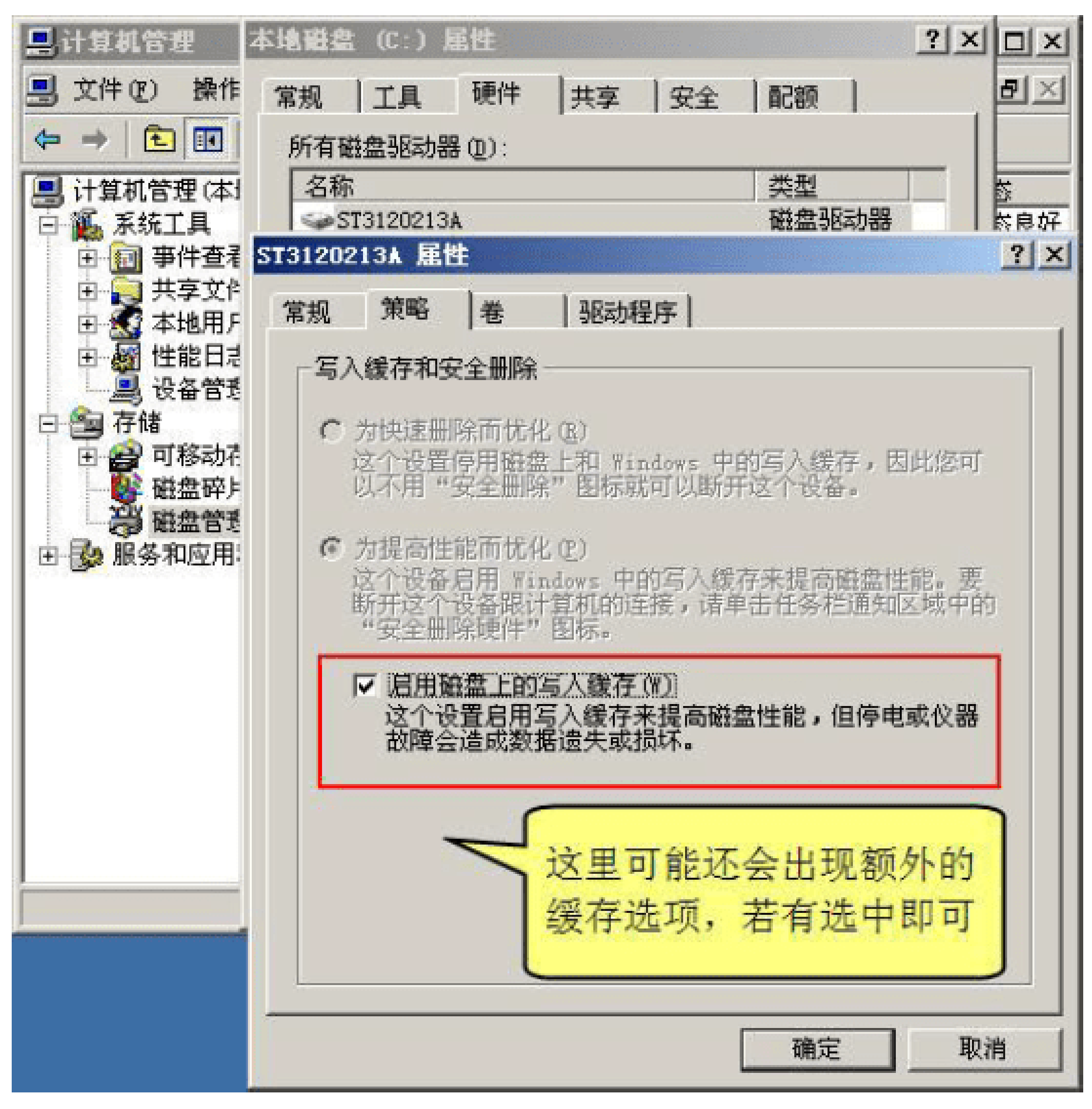


图 4-11　设置写入缓存

通过设置这样的写入缓存，笔者经过一些计算量大的程序测试，能够提高大约 5%左右的速度，也可以说是比较可观的数字。

当然，还有其他一些复杂的提高硬盘性能的方法，将在本书的后半部分性能优化的相关章节中提到。

【适合 Web 应用的硬盘小总结】

适合 Web 应用的硬盘，应该是转速高、缓存大的 SCSI 或者 SATA 硬盘，并在操作系统中开启了相应的优化功能（如果有）。

那么，类似 CPU-Z，有没有测试硬盘速度的工具软件呢？其实有很多，这里推荐一款比较流行的 HD-Tach。

4.3.5　HD-Tach 的下载、安装与使用

HD-Tach 可以到 http://www.simplisoftware.com/Public/index.php?request=HdTach 下载。它分为免费版本和商业版本。对于我们学习利用工具测试硬盘速度，免费版本已经足够了。截至 2008 年 9 月，HD-Tach 的最新免费版本是 3.0.4.0，注意它是不能用于商业用途的。

单击上述网页的 Download（下载）按钮后，将开始下载安装文件，名称为 HDTach-3-0-4-0.exe。下载完毕后双击运行，接受用户条款，保持默认设置不变，连续单击 Next 按钮，直到程序安装完成，并在桌面生成一个快捷方式，还是非常简单的。在桌面单击软件安装后生成的快捷方式，弹出界面如图 4-12 所示。



图 4-12　HD-Tach 的启动界面-模式选择

在图 4-12 中，由于所在的电脑只有一块物理硬盘，因此不用下拉硬盘列表菜单切换硬盘，同时保持默认选择的“快速检测”（Quick bench）单选框不变，单击“运行测试”（Run Test）按钮，等待一点时间，直到看到结果，如图 4-13 所示。

在图 4-13 中，红色的曲线和柱体是当前测试硬盘的实际数据。曲线表示随着文件的增大，读取速度的变化趋势；柱体表示 Burst Speed，即猝发速度，我们可以认为是硬盘的最快速度。在红色柱体上方还有 3 个灰色柱体，分别代表 ATA UltraDMA 6 硬盘、SATA150 硬盘和 SCSI Ultra320 硬盘的标准速度。可以看出，当前测试硬盘的最快速度处于比较低的水平。在图 4-13 右下角，有一个读取平均速度，值为 49.4M/s，这就是当前测试硬盘的平均读取速度，自然这个数值是越大越好。

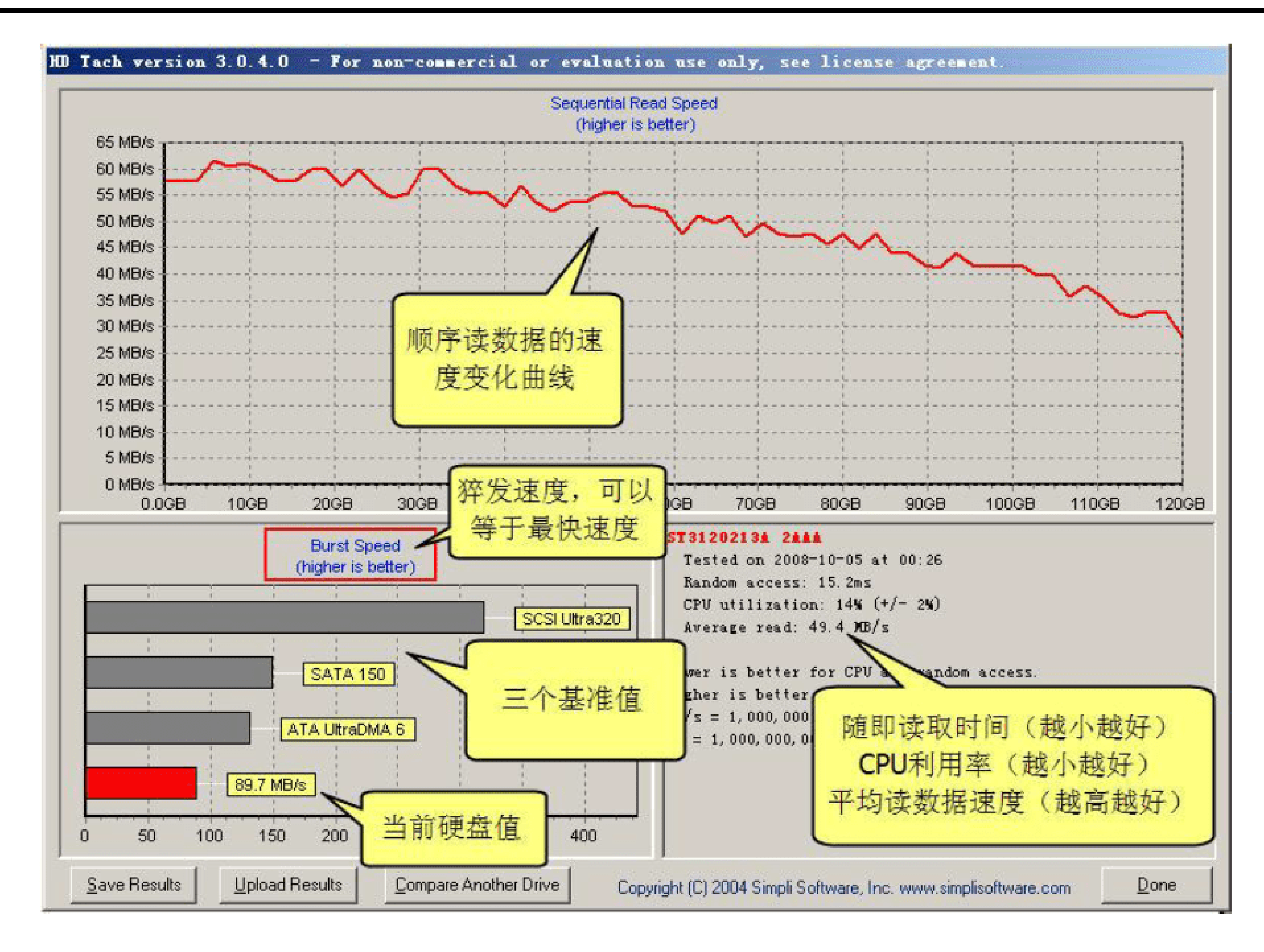


图 4-13　HD-Tach 对某硬盘的测试结果

4.3.6　Web 应用对硬盘消耗的特点

Web 应用是以用户浏览网站为基本行为的，那么，Web 应用对于服务器硬盘的消耗具有什么特点呢？小白首先回想了自己作为用户上网进行的主要活动，希望从此判断出答案。这些活动大概有以下几种。

- 浏览网页：需要下载网页上的文字内容和图片。
- 浏览视频：需要下载并播放网页上的视频文件。
- 下载软件：需要下载存放在服务器硬盘上的安装文件等。
- 发帖子：偶尔发一些帖子，大多数时间是潜水和路过。
- 上传照片和视频：频率不高，不是经常性的活动。
- 网络游戏：下载场景、进度、人物、装备等数据。
- 网络聊天：发送相对短小的文本，偶尔传送文件。如果是视频聊天，需要下载对方的图像，上传自己的图像。

从小白的网上活动就可以看出，在大部分时间，用户都需要从服务器硬盘（那些网页文字、图片、视频存放的地点）下载数据到本地。也就是说，用户对于服务器硬盘的读取操作几率要大于写入操作（上传文本、图片、其他文件等）的几率。

考虑到这样的情况，在选购 Web 应用服务器的时候，硬盘更好的读取数据能力将成为一个优势。

4.3.7　Baseline 和 Benchmark

在图 4-13 中还可以学习到两个很重要的名词。我们注意到，在图 4-13 的左下方，灰

色的柱体和测试硬盘的红色柱体（最下面的柱体）形成了鲜明、直观的对比。它们都对应一个数值，从数值的比较中也可以分出优劣。这种打分的方法叫做 Benchmark，指为了评估某计算机程序或者硬件指标的优劣，令其运行标准的测试，通过得分与已有标准结果相比较来得到结果的方法。

在图 4-13 中，参与比较的 3 个标准猝发速度，也被称为基线（Baseline），即被比较的那些标准结果。这两个词语都是从事性能测试，甚至其他类型测试工作中经常遇到的。中文名称翻译并不统一，因此有时候人们在工作场合就直接说英文名称。

除了 HD-Tach 之外，还有很多的磁盘性能乃至 I/O（即输入 Input 和输出 Output）性能测试工具，比如 IOMeter 等，这里就不多讲述了。

通过对 Web 应用性能影响比较大的两个硬件：CPU 和硬盘的选择，服务器的选择已经完成了重要步骤。对于性能测试工程师来说，了解 CPU、硬盘乃至其他服务器硬件的知识是必要的，这些基本的知识特别对于 Web 应用的性能优化很有用处。

4.4　本章小结

本章通过小白准备参加服务器选型会议的经历，介绍了 Web 性能的定义和对不同参与者的影响。

对于用户来说，网站的响应时间很重要，但是，对于网站来说更为重要。响应时间很长，将造成不好的用户体验，丧失用户对网站的忠诚度。

对于网站管理者、性能测试工程师和开发工程师来说，响应时间很长一般都意味着系统在硬件设置和软件代码方面出现了性能问题，需要他们来通力合作，找出原因，发现问题并解决问题。

在 Web 应用部署之前，选择一款合适的服务器来承载它对于 Web 性能也很重要。否则，再好的性能测试报告、性能优化建议，也会由于不合适的服务器而丧失它们应有的效果。

截至本章为止，技术部已经基本确定了承载 Web 应用的服务器配置和型号，与此同时，网站代码本身也开发的差不多了。因此，在第 5 章，小白可以开始着手性能测试的学习工作了。

第 5 章　Web 性能测试方法

良好的开始是成功的一半。小白所在公司网站的代码编写的差不多了，安放服务器的机房也已经找好，只要等第 4 章选购的服务器正式到位，就可以上机架、部署测试版本网站进行试运行了。离部署测试版的日子还有一段时间，小白对现在这段短暂的空闲，做了如下的安排：

- 首先要熟悉性能测试的几种方法。
- 在第 4 章 CPU 和硬盘的基础上，熟悉常见操作系统的性能计数器特点，并在自己的电脑上进行一次手工的性能测试。
- 熟悉常用的几种性能测试软件，听说有 Load Runner 等，从中选择一个比较好的，毕竟这也是部门经理在第 4 章开头布置的任务之一。

这 3 点内容将分别在从本章开始的第 5 章、第 6 章和第 7 章中讲述。现在就开始介绍 Web 性能测试的一些方法。

5.1　Web 性能测试的目的与方法

本节首先介绍测试的目的，然后会介绍最常见的 9 种 Web 性能测试的方法，希望读者通过这些方法，对性能测试有深度的认识。

5.1.1　Web 性能测试的目的

进行 Web 性能测试的目的很简单：

- 获得 Web 应用的性能表现情况。
- 发现并验证、修改 Web 应用中影响性能的 Bug。
- 为网站性能优化提供数据参考。

实际所做的一切测试工作都要围绕这 3 个目的来进行，这样才不会在出现这样或那样困难的时候迷失方向，导致资源浪费。

5.1.2　Web 性能测试方法的先决条件

对于性能测试方法，在某种意义上说，就是性能测试的分类。不同的性能测试分类，决定了需要采用不同的性能测试方法。Web 性能测试也是如此。

不过在介绍具体的分类或者方法之前，有必要强调一下进行 Web 性能测试的先决条件。其实，从前面的章节，我们已经能够发现，进行 Web 性能测试的先决条件有这样几条：

- 一个稳定的 Web 应用版本。该版本必须与投入生产环境的版本极其类似。这一点的原因在前文也介绍过，对一个最终不会上线运行的版本，或者功能没有完成的版本进行性能测试是没有多大意义的，除非为了演练性能测试方法本身。
- 性能测试所处的测试环境，必须独立于开发环境，并且尽量类似于实际生产环境。这一点也很重要，测试环境必须是可比较的，大致拥有相同的参照物（比如 CPU、硬盘速度等），这样的测试结果才更有参考价值。

当然，小白在今后这 3 章所做的性能测试，是为了在个人电脑上方便快速地了解性能测试方法，熟悉性能测试软件，并不打算将测试报告发送出去形成结果。对于小白和我们一般的初学者，这不失是一个好的学习方法。但是，在真正的性能测试工作中，对于上面的两个 Web 性能测试（也可以推广到所有的软件性能测试上面），一定要记牢。

下面将介绍 Web 性能测试方法的具体内容。

5.1.3　Web 性能测试的详细分类

在第 2 章中曾粗略地讲到性能测试包含性能测试方法（狭义的）、压力测试等，现在可以将绝大部分文档中提到的众多分类列举出来了，它们是如下 9 种：

- 性能测试（Performance Testing）；
- 压力测试（Stress Testing）；
- 负载测试（Load Testing）；
- 并发测试（Concurrency Testing）；
- 配置测试（Configuration Testing）；
- 耐久度测试（Endurance Testing）；
- 可靠性测试（Reliability Testing）；
- 尖峰冲击测试（Spike Testing）；
- 失败恢复测试（Failover Testing）。

看起来分类很多，但它们实际上都是为了性能测试的目的：考察应用对于系统性能的影响状况。因此，它们的区别只在于考察系统性能的角度不同。角度决定方法，这正是本节开始时将性能测试方法和性能测试分类基本划等号的原因。

5.1.4　性能测试（Performance Testing）

在介绍性能测试具体方法之前，首先说明一下性能测试（总称）和性能测试（方法）的区别。我们也分别把它们叫做广义性能测试和狭义性能测试。

【广义性能测试与狭义性能测试】

广义性能测试包括前述所有分类或方法，以考察 Web 应用对于系统性能影响状况为目的的测试活动。而狭义性能测试则是其中的一个小分类，为了区别广义性能测试中的其他分类。这种测试也是广义性能测试中最基本的方法之一。利用第一章所讲过的维恩图，广义性能测试和狭义性能测试之间的关系如图 5-1 所示。

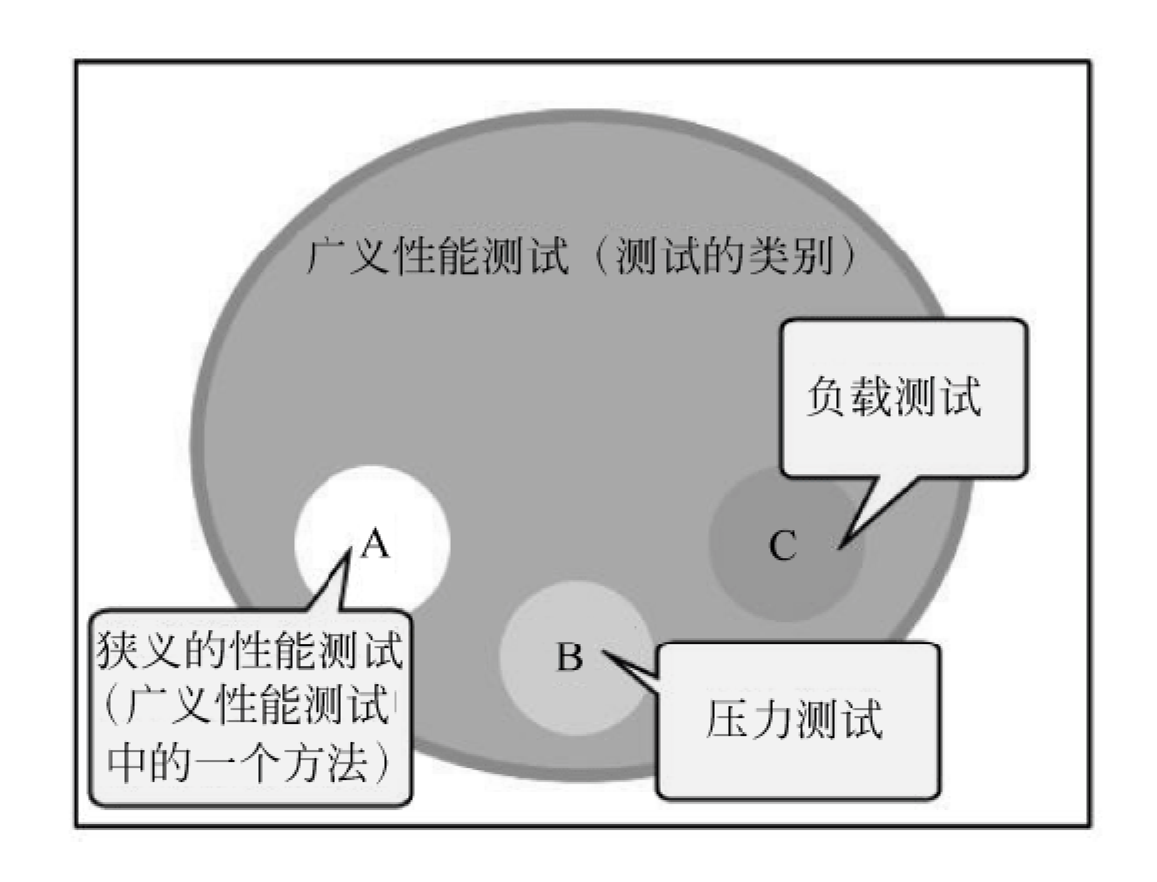


图 5-1　广义狭义性能测试的包含关系

听起来有些混淆，但对于我们要从事实际工作的人来说，名词并不重要，只需要知道有性能测试这样的总称（广义），还有性能测试这样的方法（狭义）即可。在看相关参考文献的时候，根据上下文一般能很容易判断出当前所指的性能测试是狭义还是广义。

本书中凡是“Web 性能测试”这样的字眼，都是指的广义性能测试，特此说明。

【性能测试】

性能测试（英文名称也是 Performance Testing）是这样一种方法：它通过模拟实际生产环境中运行的软件平均业务量，测试系统的性能是否满足设计说明书中的性能要求。性能测试方法在所有前述 9 种方法中是一种最基本、最常见的测试方法。这就是说，它是实施性能测试所必须进行的一种方法。

5.1.5　小白的第一次性能测试

为了尽快熟悉这种方法，小白决定利用自己电脑和公司网站的测试版本来做一次手工性能测试的实践。

一般来说，在网站尚未上线的期间内，网站的技术部门都会维护一个或几个网站的测试版本，方便开发工程师和测试工程师开展工作。即便是网站上线以后，也会继续维护这样的网站，以方便网站每日更新时的调试，待没有问题后再正式上传到生产环境中。这种结构如图 5-2 所示。

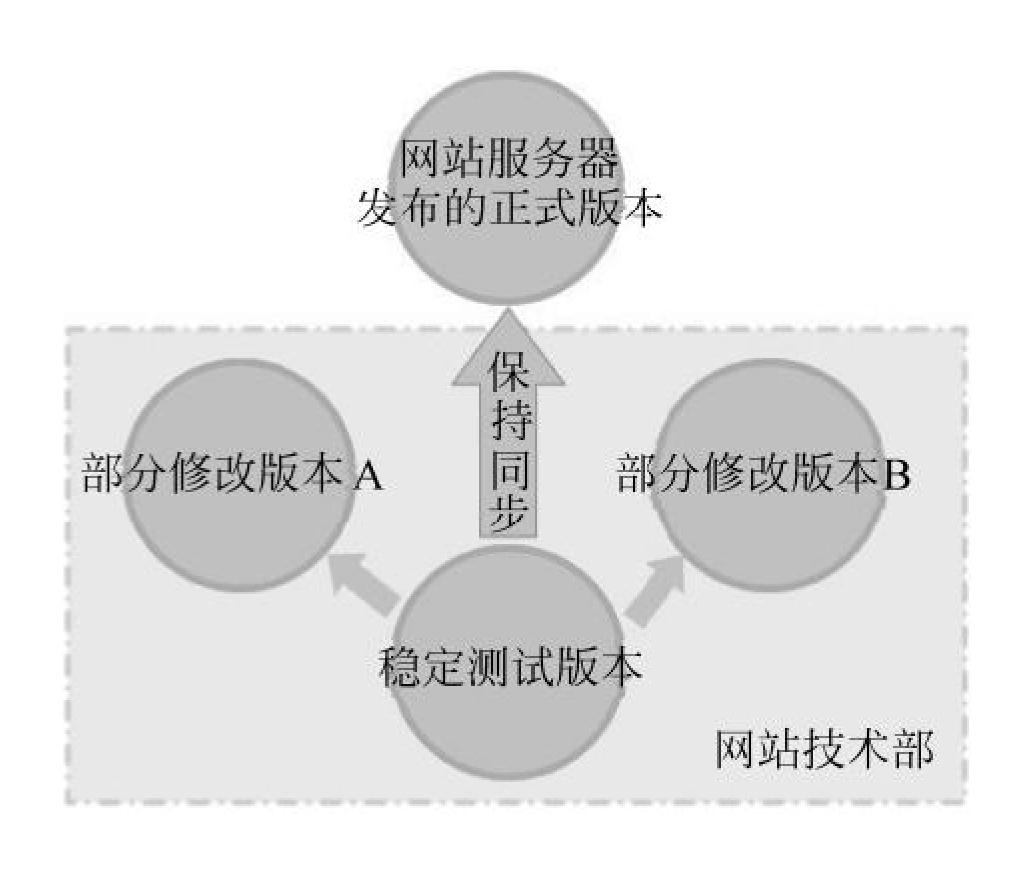


图 5-2　网站正式版本与测试版本关系示意

对于小白的第一次性能测试，该如何选择测试所针对的网站版本呢？有两点需要考虑：

- 由于网站正式版本尚未发布，目前还没有正式版本可供选择。
- 即使正式版本发布也不能直接使用它来测试（因为这样会影响用户的使用）。

根据以上两点，选择图 5-2 中的稳定测试版本。

截至目前为止，小白所知道的仅仅是打开网站的响应时间，所以他想从这个时间入手，来评估一下网站测试版本的性能。他是这样进行测试的：

- 打开浏览器，输入网站测试版本的网址，按下 Enter 键，并记录此时的机器时间。
- 在网站全部打开后再记录一次当前机器时间。
- 两者相减，得到一个时间间隔。
- 重复多次，最后计算平均时间间隔。
- 对比网站的设计说明书，在那里一般都会有以响应时间为标准的性能要求。
- 根据两者数值的比较，决定网站测试版本的性能好坏。

【有关时间的记录】

在性能测试过程中，往往免不了进行当前时间的记录，或者计算两个时间的差值。由于 Windows 系统任务栏中的时间数字较小，读者可以在互联网中寻找一些显示桌面时钟的免费软件（显示时间最好可以包含秒数以达到精度要求）。当然，这样的时间记录是依赖于肉眼，精度依然不够高。可以通过编程的方法来获得更精确的时间。

虽然小白所列出的上述测试过程看起来与用户浏览网站没有什么不同，而且，也看不到使用哪些高级的工具软件，但它确实是一次人工的性能测试。性能测试真的都是这么简单吗？小白非常好学，进而思考了下列几个问题。

5.1.6 小白的思考

在第 3 章中我们已经知道响应时间，并且知道用户所感觉的响应时间并不很准确。那么，小白这次测试应用了计算机时间等“高科技”计时装备，是否就意味着准确呢？

1. 响应时间的问题

其实，对于感觉和记时两种方法，响应时间数值都是差不多的。那么，访问 Web 应用，多长的响应时间说明性能比较好呢？实际上依赖于几条标准。

- 软件设计说明书中的标准：根据用户的需求，一般都会列出。
- 不成文的习惯标准：如果在设计说明书中没有列出，那么可以参考国外的业内公认标准，即 3/5/10 原则。
  - 在 3 秒钟之内，页面给予用户响应并有所显示被认为是“不错的”。
  - 在 3～5 秒钟内，页面给予用户响应并有所显示被认为是“好的”。
  - 而 5～10 秒钟是可以“勉强接受的”。
  - 超过 10 秒钟就有点让人不耐烦了，用户很可能不会继续等待下去。
- 在尽可能合理的情况下，响应时间应该越快越好。

另外，响应时间包含了网络传输数据的时间、DNS 记录查找时间和真正由网站服务器处理的时间，因此，遇到时间间隔很长的情况时，首先要排除前两个时间的影响。

另外，还有很重要的两点不能忽略：

- 小白只是以一个用户的身份去访问网站的测试版本，而网站一旦投入使用，真实情况是会有上万人同时访问它，那么响应时间还会有现在这么好吗？
- 小白是在公司内部进行测试的，要知道公司内部的局域网一般都是百兆、千兆网，速度非常快；如果换到家里，用 ADSL 之类的上网条件，响应时间还会如此快吗？

这几个问题都说明小白的这次性能测试确实欠缺很多因素。不过，这正是我们在下面的章节要学习的。

2. 测试场所和指标的问题

小白在进行测试的时候，记录的是自己电脑上的时间间隔，从它数值的大小来间接判断服务器端性能的好坏。那么，能不能直接获得服务器端的性能数据，岂不是更加精确吗？

是的，完全可以。响应时间所带给人的只是性能好坏的大概印象，如果要更加专业的测试性能，需要获取服务器端的指标数据，我们管这些指标叫做性能计数器（Performance Counter），在第 6 章，我们将重点介绍它们的单个含义以及获取方法。

综上所述，小白基于目前理解的第一次性能测试有了结果，虽然过程远远不够，但也让我们体会到了性能测试所关注的要点，进行的大致过程。简单地说，Web 应用的性能测试方法，就是通过模拟若干用户对于网站的访问，获得性能计数器和其他指标的数据，再分析它们以进行性能评估，使得关注性能测试的各方对系统性能有基本的认识。

5.1.7　压力测试（Stress Testing）

相对于前面性能测试方法的普通，压力测试（Stress Testing）方法可以说走了一个极端。它测试 Web 应用在事先规定的某种饱和状态下，比如 CPU 处于 75%利用率的情况下，系统是否还具备处理业务的能力，或者系统会发生什么样的状况（出现错误？系统宕机？等）。

一句话，压力测试是考验一个系统的抗压能力的：在当前比较大的压力下，它能否承受得住。压力测试的目的是为了测试 Web 应用的稳定性。

【压力测试与体操比赛】

在体育比赛场上我们可以看到生活中的压力测试，例如体操比赛中的规定动作环节。场上选手在比赛时，其动作组合必须包含组委会所设定的所有规定动作，如图 5-3 所示的经典规定动作——托马斯全旋。通过在这样的条件下比赛，裁判来考察运动员的完成质量，由于动作难度系数基本一致，重点将是完成质量的稳定性。通过这个类比，压力测试就很好理解了。



图 5-3　类似压力测试的体操规定动作比赛（图中动作为托马斯全旋）

压力测试方法有如下的两个特点：

（1）压力测试方法的目的是测试系统（本书中为 Web 应用）的稳定性。人们对很多软件系统都有这样一个经验：当系统处于较大压力的时候，如果还能够维持正常工作，那么，就能说明它在压力不大的一般条件下，具有长时间正常工作的能力。从这里可以看

出，压力测试方法有一点“一叶知秋”、“以小见大”的含义在其中。

（2）压力测试方法的具体操作过程是通过对系统施加负荷（模拟用户对 Web 应用的访问等），使系统的资源占用保持在一个事先约定的水平（比如前文所提到的 CPU 占用率 75%），来检验此时系统的表现。测试的重点在于系统对于用户的响应时间变化、系统是否出现错误甚至崩溃等。

5.1.8　负载测试（Stress Testing）简介

在实际工作中，负载测试方法和压力测试方法往往被放在一起谈论，因此很容易混淆，其实它们的区别是很明显的。

【负载测试方法】

负载测试（Load Testing）方法通过在被测试系统上不断增加负荷，直到事先选定的性能指标（比如响应时间），变为不可接受或系统的某类资源使用已经达到饱和状态。负载测试方法实际就是一个不断加压，直到找到系统不可用临界点的过程，形象地说，那一点正是“强弩之末”。

【负载测试方法与举重比赛】

在 5.1.7 节我们把压力测试和体操比赛的规定动作相类比，在这里我们将负载测试方法类比为举重比赛，如图 5-4 所示。在比赛中，选手不断地增加重量，挑战自己的极限，直到杠铃加到某一个重量时，3 次试举都失败。这一重量就是举重比赛的最终结果。

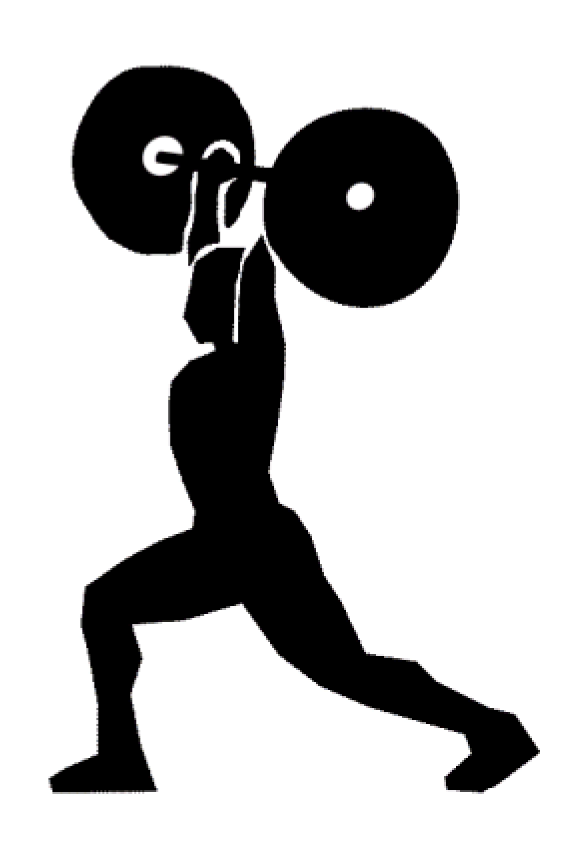


图 5-4　举重比赛与负载测试有相同之处

通过负载测试方法，我们可以发现系统的处理极限点在哪里。

5.1.9　负载测试的特点

负载测试方法有如下几个特点。

（1）它的主要目的在于找到系统处理能力的极限，为系统进一步优化做参考。另外，这种测试也可以用来比较不同的优化方法对于性能极限的提升，因此也可以称之为可扩展性测试（Scalability Testing）。这个名词可以用图 5-5 清晰地表述出来。

在图 5-5 中，2 条曲线分别代表两种优化方法经历负载测试的结果。A 方法的性能极

限在 A 点，B 方法的性能极限在 B 点。根据负载测试的定义，比 A、B 两点值小的部分都是系统的安全运行区间。由于 B 的数值要大于 A，说明采用 B 方法优化，系统的可扩展性提高了。

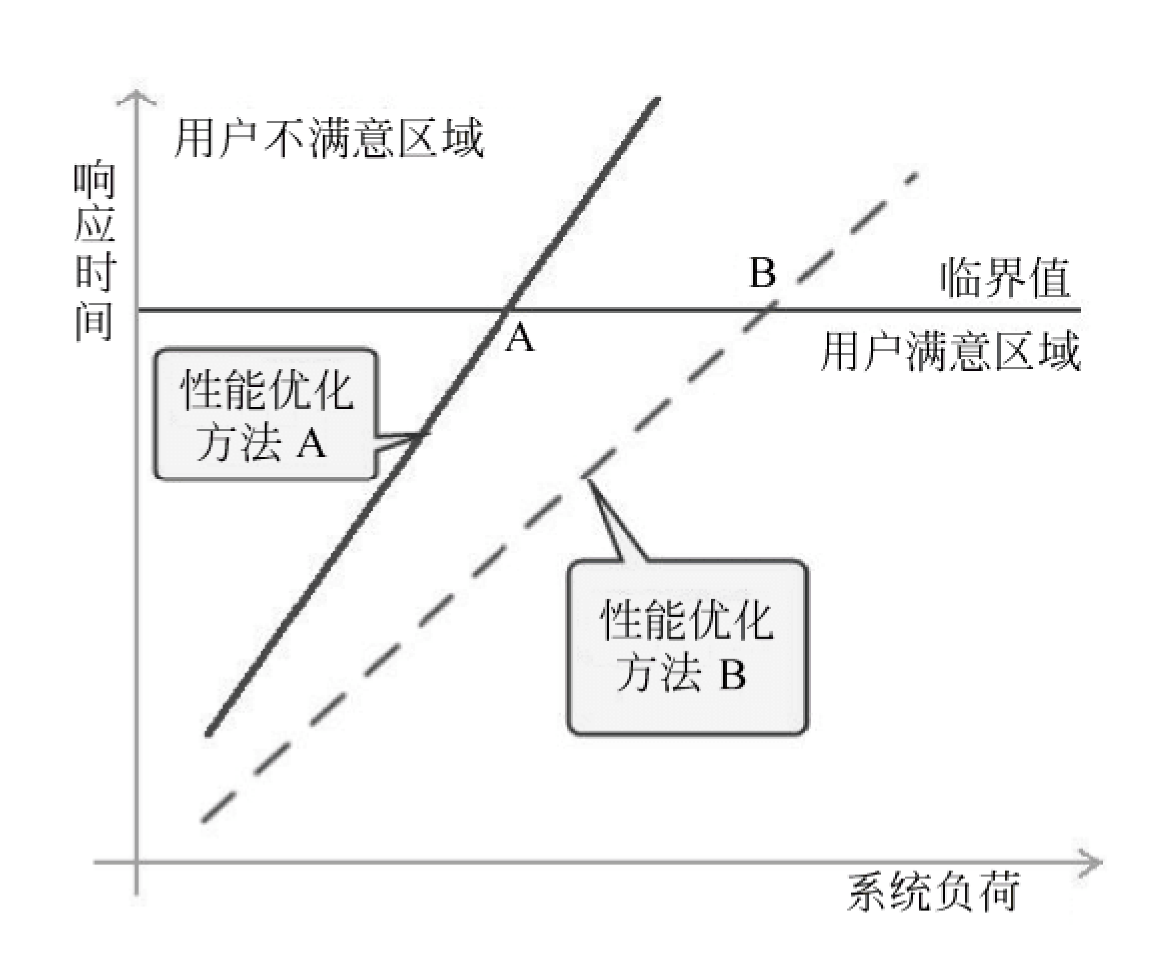


图 5-5　负载测试用于优化方法的比较：B 好于 A

（2）负载测试方法的操作是一个不断加压的过程。负载测试方法是一个“性能指标记录——增加负荷”的操作循环，直到预定被关注的性能指标不再令人满意。这个极限点在测试结果中的表示类似这样的形式：“在给定条件下当前 Web 应用将最多允许 10000 个并发用户访问”、“在给定条件下当前 Web 应用最多能够在 1 分钟内处理 1000 次用户对数据库的修改”等。常见的在负载测试方法中被关注的性能指标包括：Web 应用的响应时间、Web 服务器平均 CPU 利用率等，它们的具体数值需要根据实际情况来调整。

（3）负载测试方法要考虑被测 Web 应用的实际业务负荷量与正确的使用场景，以保证测试结果具有参考价值。

【实战演练：教训】

在这方面，笔者的同事曾经有一个教训。有一个网站，可以通过 Web 直接访问，也可以通过 RSS 进行订阅。在网站发布之前，网站技术部门的所有工程师都认为绝大部分用户都是通过 Web 来访问的，因此，在时间紧迫的情况下，重点测试了 Web 访问的性能，对于 RSS 相关代码测试的就很少。结果在网站上线之后，他们惊奇地发现，大部分用户访问都是通过 RSS 来完成的，因为负载测试做的很简略，结果每过多久服务器就被拖的几乎无法访问了。可见，对于负载测试，乃至整个性能测试而言，模拟真实的应用场景是多么的关键。

5.1.10　并发测试（Concurrency Testing）简介

并发测试（Concurrency Testing）方法通过模拟很多用户在同一时刻访问系统或对系统的一个功能进行操作，来测试系统的性能，从中发现问题。网站就是一个很典型的需要并发测试的应用场景：当网站运行的时候，在世界各地同一时刻都可能有很多用户在进行同一个操作，如图 5-6 所示。除此之外，类似的还有银行的某些系统、航空的某些系统等，它们都需要大量用户同时访问和操作。

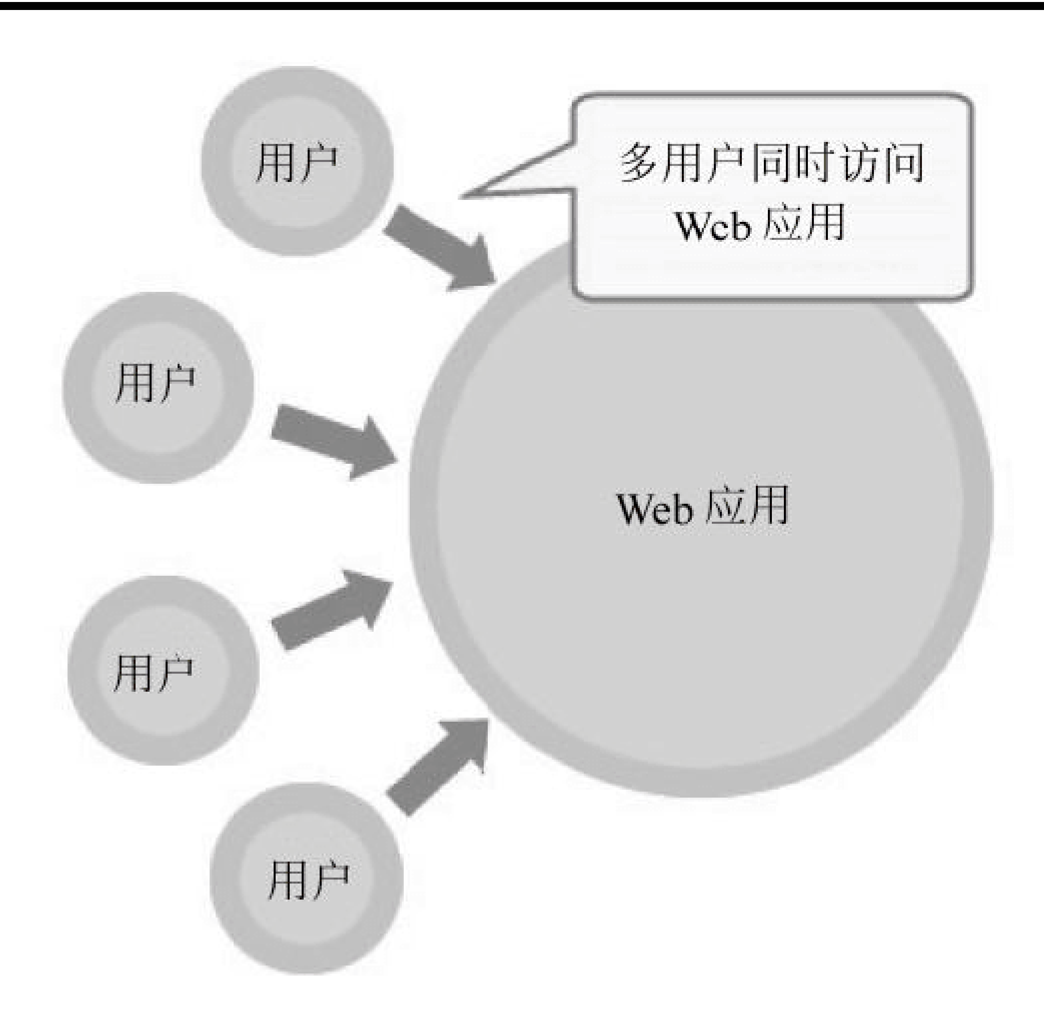


图 5-6　用户对 Web 应用的并发访问

类似小白在本章开始对公司测试网站的访问是无法发现可能存在的并发问题的。即使是公司所有的同事一声令下，也无法做到比较精确的同时访问，数量也很不够。因此，并发测试是无法用人工的方式来完成的，必须依赖一些工具软件来模拟实现，比如 HP 公司的 LoadRunner 等，这个软件我们将在后面的章节专门介绍使用方法。

并发测试所要考察的是系统在并发处理方面是否存在缺陷。实现并发的编码与原理是比较复杂的，需要很高级的开发技巧，因此不在本书介绍的范围之内。在实际工作中，我们只需要了解如下 3 个要点：

- 并发测试需要用工具模拟多用户的访问。
- 要熟悉第一点提到的工具测试并发的操作。本书介绍的是 LoadRunner。
- 要了解并发测试所关注的性能问题是什么。

至于并发中深入的线程、进程、内存泄露等知识，则需要在工作中遇到问题时来学习了。

5.1.11　并发测试所关注的性能问题

并发测试关注哪些性能上的问题？为了理解方便，我们用一个通俗的例子来说明。

还是利用传统的类比——饭馆，毕竟民以食为天。在某一时刻的饭馆中，有很多客人都在吃饭，这可以说是一个并发场景。于是，可能出现的问题有如下几个（当然不限于以下的这些）：

- 餐桌是有限的，客人很多，超出桌子数目，只能排队叫号了。类似这样抢空闲桌子的情况在程序代码中也会出现，叫做资源（内存等也被称为资源，程序运行需要在内存中，而可用的内存又是有限的）的争用（Race）。
- 如果饭馆还有一个空桌子，但被别人电话预定了，而此时饭馆等位的人又很多，服务员该如何处理呢？一般是等待 15 分钟、联系预定人的电话等方法。类似这样有关保留预定与放弃预定的情况在程序代码或者数据库中也会出现，根据条件不同（比较复杂，涉及类似多个客户订同一张桌子等细节），分别叫做活锁（livelock）和死锁（deadlock）等。

- 如果饭馆的服务员应付众多的客人已经忙不过来；或者就是服务员忘记了，经常会出现这样的情况：有一桌客人在吃完饭后已经结账走人，但服务员并没有及时的清理桌布，收盘子、擦桌子等，导致后面的客人无法使用这张桌子进餐，无形中使得饭馆的接待能力下降。由于代码的问题使得类似上面这样的情况在程序运行中也出现，就叫做内存泄露（Memory leak）。

以上几个类比可能不很确切，但是引申出了一个结论：并发测试所关注的性能问题就是：系统中的内存泄露、线程控制（锁的问题）和资源争用。

5.1.12　并发测试的特点与工具

并发测试具备如下的几个特点：

（1）并发测试可以是黑盒测试，也可以是白盒测试。测试工程师可以不了解代码实现的细节，通过工具软件实施并发测试找出 Web 应用的并发问题。开发工程师也可以通过并发测试对自己编写的代码做单元测试。

（2）并发测试可以在项目进行的大部分时候进行。在项目的早期，它可以通过结果大致验证系统总体设计和结构是否合理；在项目编码阶段，它可以发现代码的并发问题；在项目的测试阶段，它可以发现整个系统的并发问题。

【并发测试工具】

除了前文提到，本书后面章节要介绍的综合性能测试软件 LoadRunner 之外，还有很多专用的并发测试工具，比如在 Java 平台下有 JProfile、JProbe 等；在.NET 平台下有 CHESS、Zing 等。

由于并发测试这部分内容程度比较深，完全展开需要更多的是开发知识，而不是测试知识本身，感兴趣的读者可阅读相关的书籍。

5.1.13　配置测试（Configuration Testing）

所谓配置测试（Configuration Testing）方法，是通过对被测系统所处的软、硬件环境进行设置上的调整，来了解其对于系统性能影响的程度，并根据结果发现环境的最优配置组合。这个测试方法主要用于性能的优化，一般用于 Web 应用正式投入使用前夕和运行当中。

1．配置测试的实例

实际上，在使用电脑的过程中，我们每个人都可能做过这样的测试。比如，使用 Windows XP 一段时间后，电脑运行速度可能有所减慢。那么我们可能就会上网查询具体变慢的原因，更改一些系统默认的设置，并从实际的效果来验证这些设置的更改是否有效。无效的配置很可能被恢复成默认值。这可以说就是一种配置测试。

2．配置测试的目的

配置测试的目的就在于发现当前修改的这种配置是否能够有效提高 Web 应用的性能。还记得有一种比较流行的工具软件：Windows 优化大师吗？它实际上就是通过调整不

同的系统软、硬件参数，使得我们的 Windows 运行起来感觉更快。Windows 优化大师的软件界面如图 5-7 所示，可以发现它是由多个配置修改页面组成的。

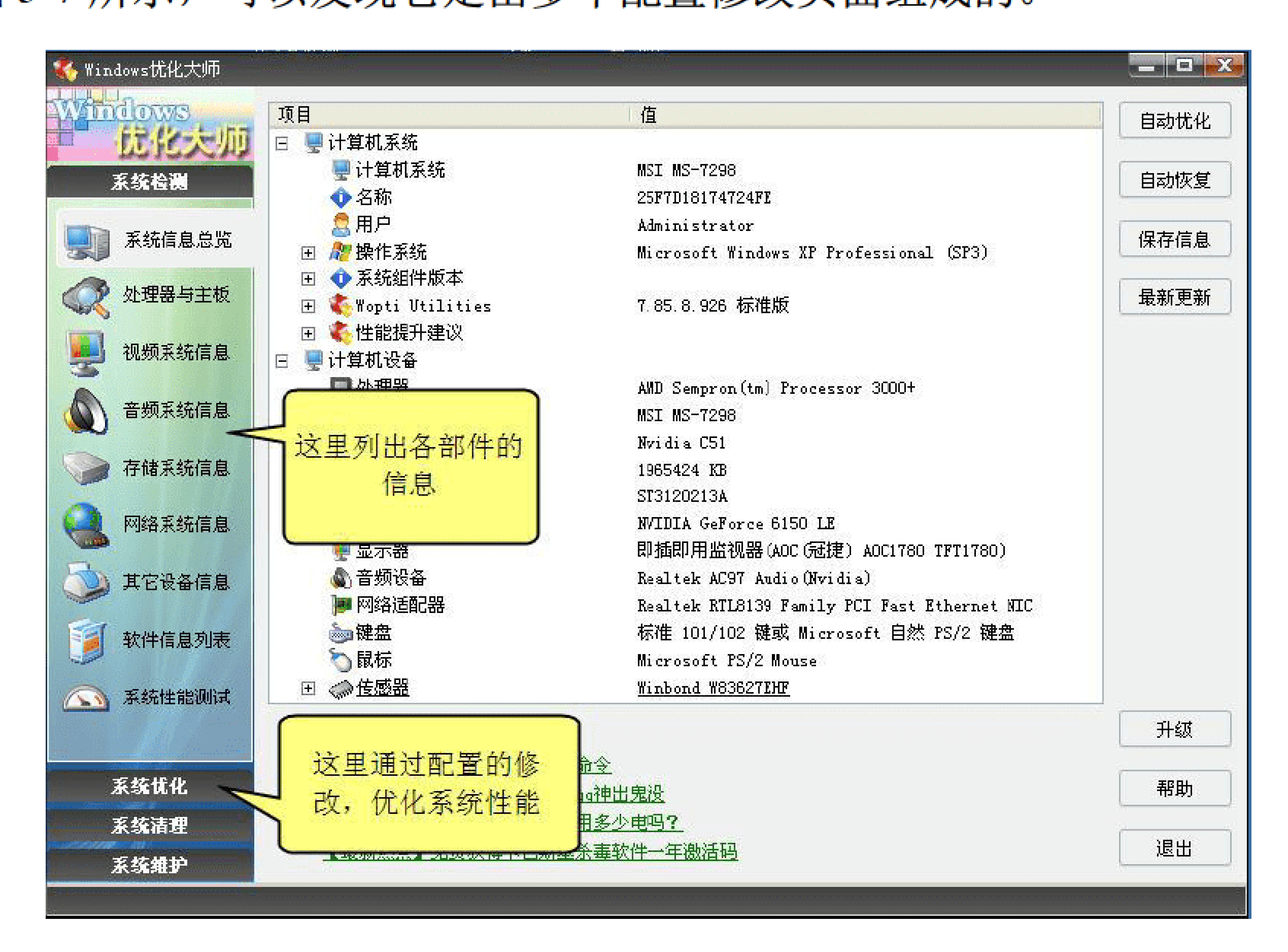


图 5-7　Windows 优化大师的界面

3．配置测试实施的时机

那么，什么时候进行配置测试呢？还是与我们平时使用电脑的情况做类比。当我们尚未把所有需要的软件都安装完毕之前，一般是不会做配置测试的，这是因为这段时间即使修改了软、硬件配置、进行了优化，这种配置也可能被新安装的软件在之后覆盖掉，导致优化失效。同样的道理，在 Web 应用的程序代码没有开发完毕、测试没有基本完成、有关性能的 Bug 还远远没有被修改的时候，就进行配置测试、性能优化是不合适的。也就是说，配置测试测试的是 Web 应用所依赖的软、硬件配置对于性能的影响，对于 Web 应用的代码本身，已经假设它达到了最好的性能。

配置测试所涉及的系统设置要依照 Web 应用所依赖的环境而定，一般分为软件和硬件两部分：

- 软件部分：数据库各参数的设置；操作系统各参数的设置；网络带宽的设置等。
- 硬件部分：硬盘缓存、硬盘运行模式、磁盘阵列的设置等。

5.1.14　耐久度测试（Endurance Testing）

耐久度测试又叫做浸泡测试（Soak Testing），具体方法是令被测试的软件系统、Web 应用在大负荷条件下长时间运行，从中发现问题。从这个定义来看，被测软件系统或者 Web 应用长时间处于测试状态下，用“浸泡”来描述是很恰当的。

耐久度测试所能发现的问题都和被测系统运行时间变长后，一些资源无法释放，导致系统响应时间慢慢变长有关。详细而言，有以下几类：

- 严重的内存泄露（请见并发测试小节）导致系统内存慢慢不够使用。

- 数据库连接、数据库游标、应用服务器资源等没有适时释放，导致系统变慢。
- 被测系统代码中的数据结构不甚健壮或合理，在长时间运行后，对其的增加、删除、修改、查询等速度出现问题。

耐久度测试需要至少关注以下一些指标：CPU 使用率、可用内存、内存使用百分比等。通过隔一段时间记录以上的指标，最终形成数据表和相应的图，从中可以找到规律，如图 5-8 所示，它是在进行一次耐久度测试时，每小时在线用户数量的结果绘图。

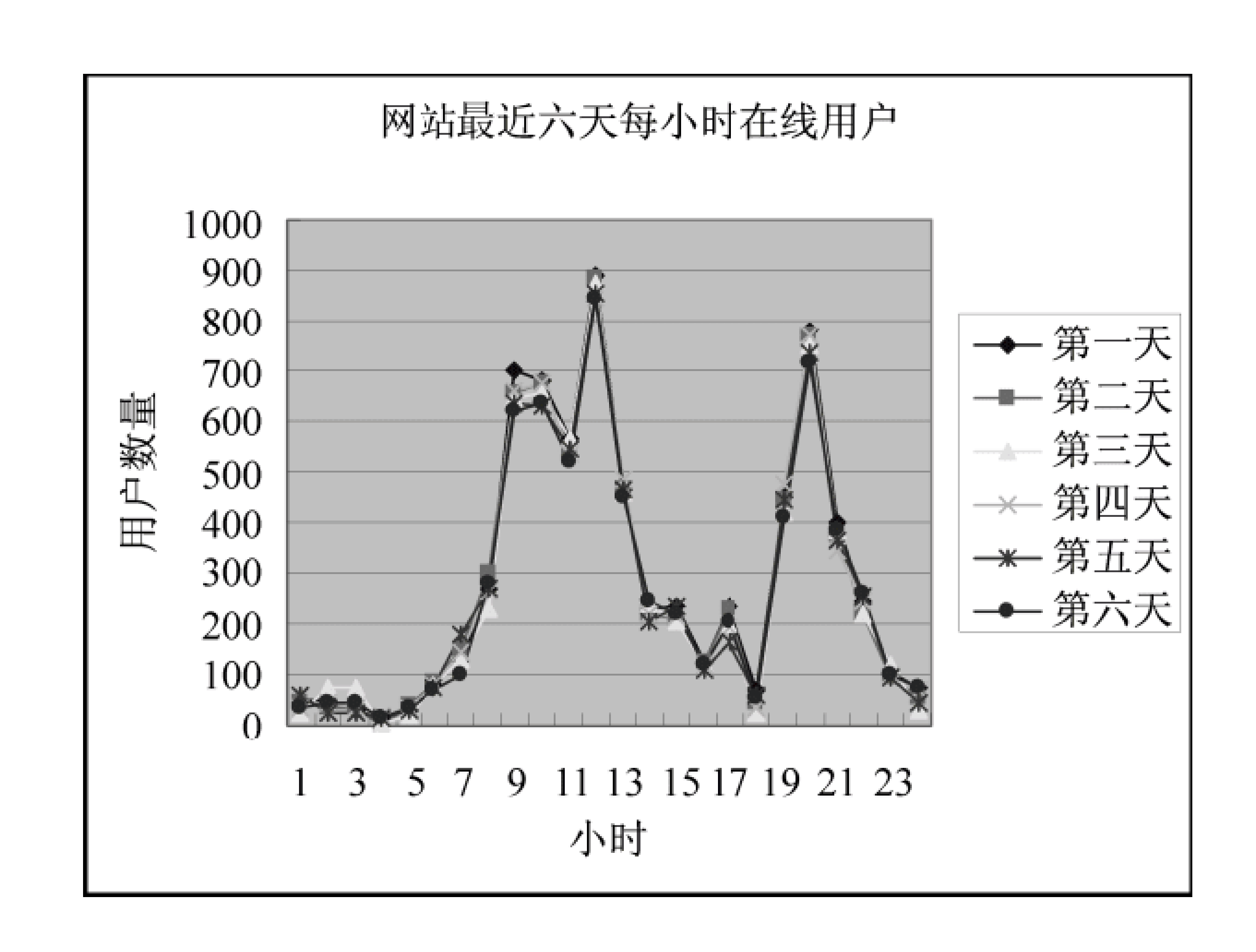


图 5-8　某网站最近 6 天每小时在线用户统计

根据图 5-8 的变化规律，再结合耐久度测试中定时记录的 CPU、内存等指标，如果二者规律不符合（比如当在线用户数少的时候，内存占用并没有下降很多），就可以分析出有关资源分配是否正常的结论。

耐久度测试可以在代码开发阶段，也可以在网站版本接近完成、准备上线前，更可以在网站运行当中。因此，根据这几种情况，进行耐久度测试的时间长度有所不同，但总的原则都是测试时间要尽可能地长，这样才有“浸泡”的作用。

- 代码开发阶段，主要看开发工作的时间安排与发现问题的早晚。如果运行早期就发现了内存泄露问题，可以中断进行代码的修改。
- 网站版本接近完成的时候，主要看网站上线时间安排，和发现问题的大小。如果上线时间不变，则耐久度测试进行到上线为止。
- 网站运行当中的耐久度测试，由于要定时记录各种指标，对于网站本身可能影响较大，需要择机进行，比如在长假、长周末等时间。

【耐久度测试与其他测试的区别】

耐久度测试主要考虑的是时间对于系统或者 Web 应用的影响，因此，它测试的时间要比其他方法，如性能、压力、负载等测试要长得多。另外，它关注的是系统在一个渐进的资源消耗过程中的表现，与压力测试关注一个固定指标下（可以看作一个时间点）系统的表现、负载测试关注最终的那个最大负载（也可以看作一个时间点）、并发测试关注并发操作发生时系统的表现（也可以看作一个时间点）都不同。

5.1.15　可靠性测试（Reliability Testing）

可靠性测试（Reliability Testing）方法实际上就是前一节所说的耐久度测试，只是这个词一般用于测试大型软件，特别是应用于工业、交通等的行业软件。这是因为 IT 领域的可靠性测试这个词是从制造业“引用”过来的，带有些许传统大工业机器制造的意味。

5.1.16　尖峰冲击测试（Spike Testing）

与可靠性测试类似，尖峰冲击测试这种方法也是从其他行业借鉴而来。在电力工业，有一种冲击测试，用来验证设备在刚刚接通电源时能否经受住涌流的破坏。所谓涌流，通俗地说，就是电源接通瞬间，电流突然变大的现象。涌流过后，电流逐渐恢复到正常的水平。

软件行业的冲击测试，或者说本书称之的尖峰冲击测试，就是为了验证网站在用户突然极具增加的情况下能够正常工作。我们知道，在网站的运行过程中，会经常出现各种各样用户数量的突然增加：

- 网站开幕时可能导致用户急剧增加，超过预期。
- 网站公布与用户极为相关的信息，比如高考成绩、录取分数等。
- 网站投放一些商业促销广告和促销活动，比如季节性降价，春节前大促销。
- 网站举办酝酿已久的明星访谈、在线销售演出、比赛门票等吸引眼球的活动。

以上这些情况产生的在线用户数量突然增加都会对网站性能产生巨大影响，读者一定记得通过网络购买奥运会门票时，由于用户非常踊跃，导致售票网站无法打开的案例。

在前文我们介绍过负载测试，但实际情况所产生的负载不会老老实实地遵循最大负载的限制，很可能在短时间内就会超过，这时系统并不一定会出现问题。尖峰冲击测试就是为了验证此时网站的应付能力。

如图 5-9 所示为网站在某一时刻，在线用户突然增大，形成一个尖峰的情况。这也正是尖峰冲击测试中 Spike 的由来，Spike 在英文中是钉子的意思。

【尖峰冲击测试的实施】

尖峰冲击测试一般也是采用工具软件进行自动测试的。在 Load Runner 中，可以修改之前性能测试的脚本，令某一个时刻用户数突然增大，就可以达到测试的目的。

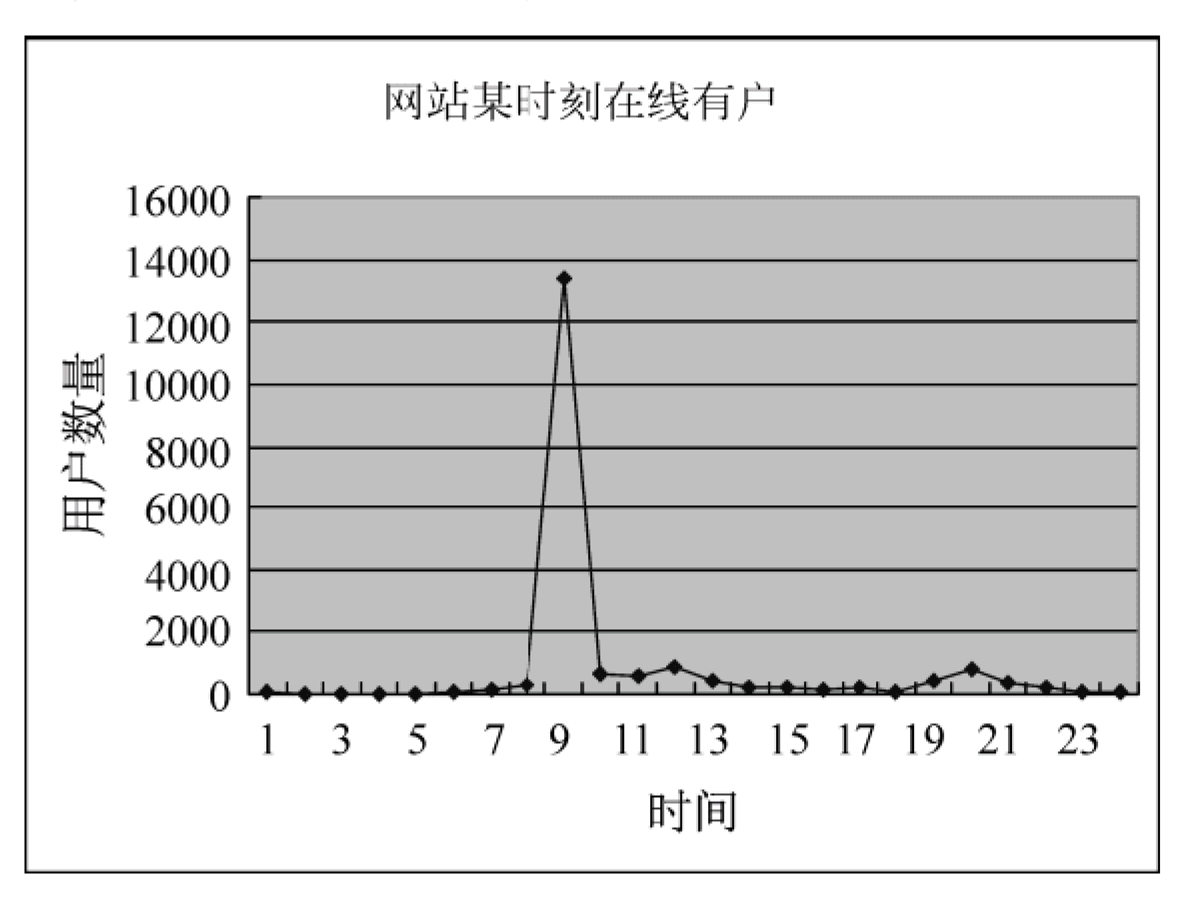


图 5-9　网站某时刻在线用户数突然增大形成尖峰

5.1.17　失败恢复测试（FailOver Testing）

失败恢复测试（Failover Testing）方法对于大中型的 Web 应用是很重要的，它针对有冗余备份（Redundant Backup）、负载均衡（Load Balance）的系统。这种测试方法用于验证某部分 Web 应用发生故障时，整个网站是否能够继续让用户使用的能力。

1. 用户访问网站可能出现的问题

我们知道，Web 应用是存放在服务器的硬盘上供用户访问的，而服务器一般来说都位于 IDC（互联网数据中心，Internet Data Center）机房的机架上。用户访问一个 Web 应用的过程如图 5-10 所示，为简单起见，这里只列出大的部分。

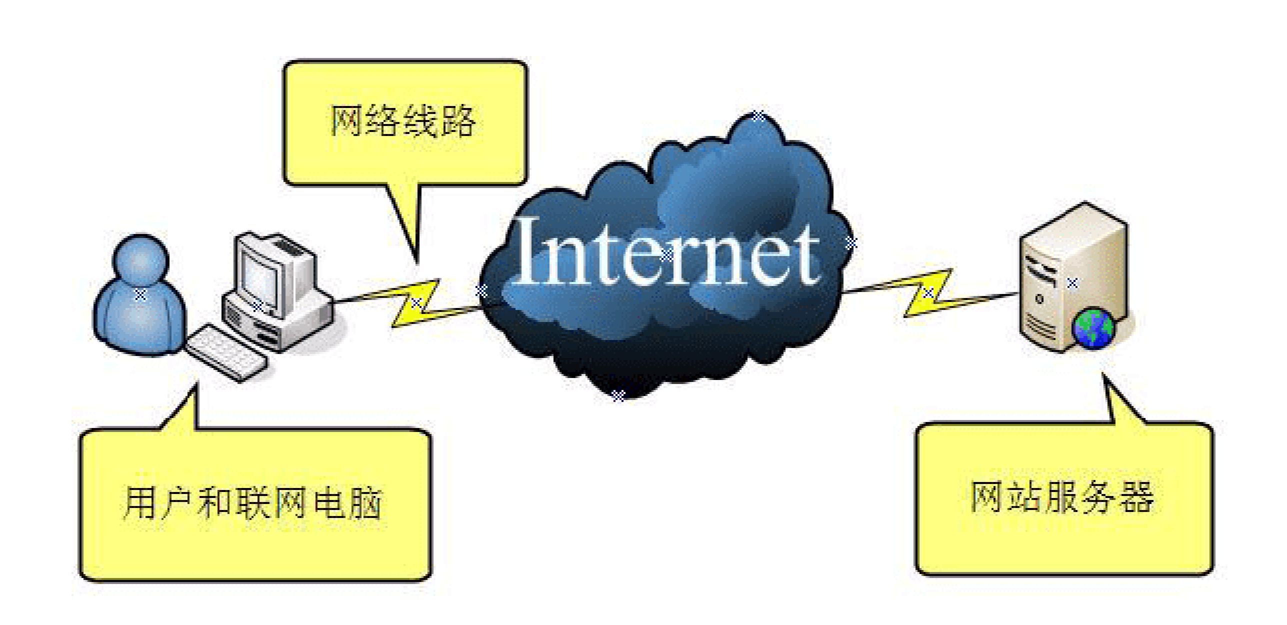


图 5-10　用户访问网站过程的示意图

从图 5-10 中可以看出，如果一个用户无法访问某个网站，那么可能是：

- 用户电脑出现了问题。解决方法：维修用户电脑。
- 网络线路出现了问题。解决办法：更换网线，修正交换机、路由器等设置。
- 网站服务器出现了问题。解决办法需要在下文进一步讨论。

那么，如何使得网站具有高可用性呢？

【何为高可用性】

所谓高可用性，就是指网站在绝大多数的时间都能够正常显示。网站的设计说明书中经常提到“保证网站 99.99%的时间都可用”这样的承诺，这就需要一定的技术保障。但是 Web 应用比较复杂，软件代码可能有 Bug；用户的持续访问和数量的增加也会造成网站较大的负荷，导致硬件可能失效。

可以用以下几种方法来提高网站能正常运转的时间。

- 主动防御：尽量修改软件代码中的 Bug，提高 Web 应用本身的健壮性，减少网站出问题的机会。
- 被动防御：在保持现有 Web 应用代码、服务器配置不变的情况下，通过适当增加服务器的数量、尽量平均分配网站负荷、采用备份的方法提高可用性。在这些措施中，就有冗余备份，负载均衡等方法，我们将在下面的内容中介绍。

2. 提高可用性：冗余备份

冗余备份是用两台或者多台服务器构成一个小的服务器场（Server Farm）来承担网站

失败恢复工作的。简单地说，一台服务器作为 Web 应用的主服务器，另一台作为它一模一样的镜像复制。一旦主服务器发生问题，网站无法正常使用，立刻切换到备用服务器，使得用户的访问能够继续。它的结构示意如图 5-11 所示。

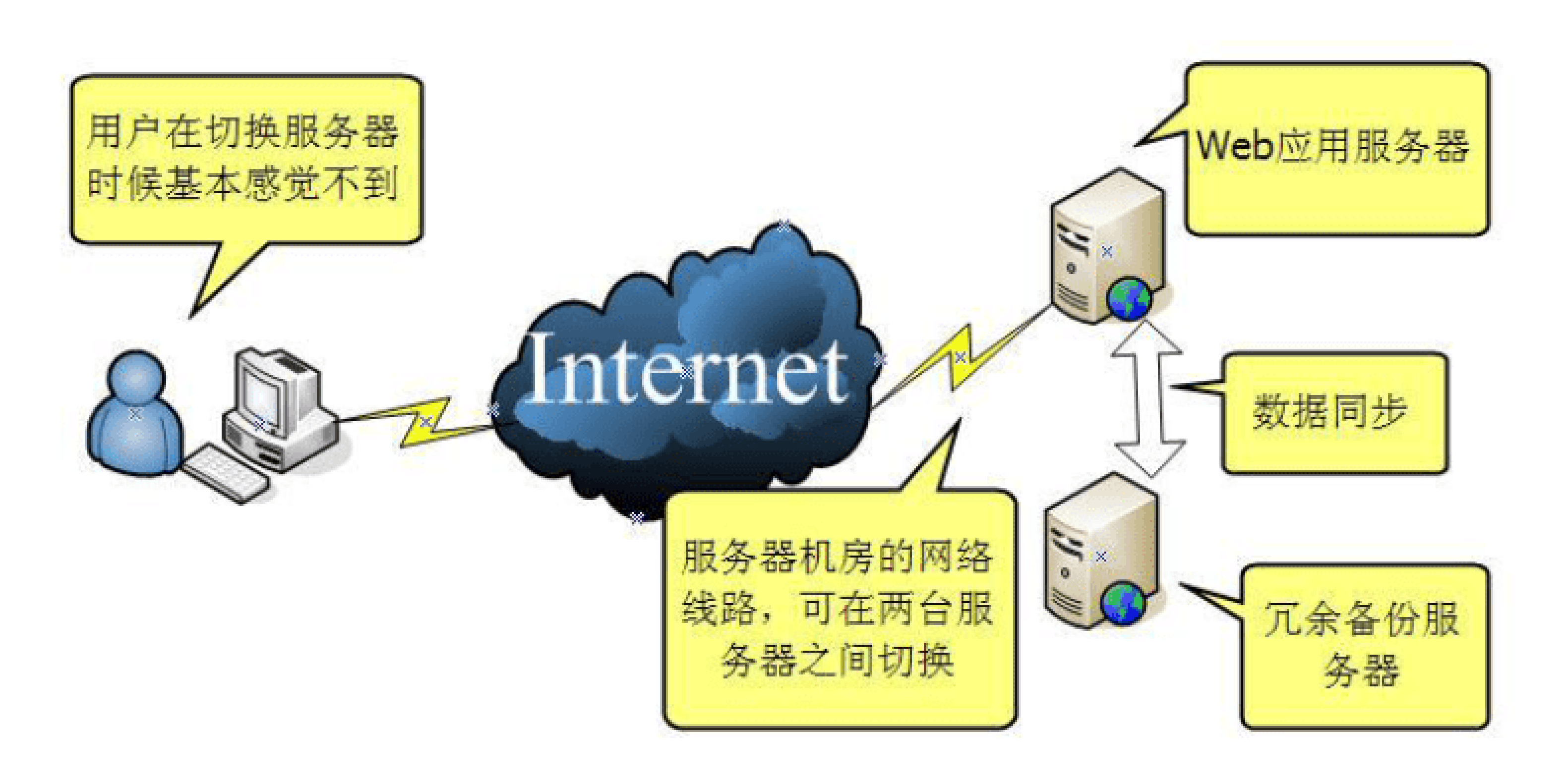


图 5-11　冗余备份的简单结构示意图

由于网站所需要的服务器要比图 5-10 中多出一台到多台，这些服务器中的数据又是互为备份的，因此，我们把这种方式叫做冗余备份。当图 5-10 中的 Web 应用服务器发生故障时，系统检测到（有多种方法能检测服务器故障是否发生：比如“心跳”，即 HeartBeat，采用定时 ping 服务器以及其他指标），系统将把连接服务器与外界的线路自动转接到备份服务器上，从而导致用户端浏览网站时基本感觉不到这种变化，极大地减少了维修 Web 应用服务器所带来的网站停止时间。

这种方法在生活中也经常采用。比如家里的手机、电视、汽车等出了故障，维修点一般会先给一个替代用品来满足维修期间的需要。不同的是，对于网站来说，主服务器和备份服务器的数据要求尽量完全一致。

3．提高可用性：负载均衡

冗余备份看似很完美地解决了服务器故障所带来的影响，但是它增加了设备，增加了成本，而且备份服务器上网站的备份实际上用户也是可以使用的，平时闲置的话有些可惜。因此，人们又采用了负载均衡的办法，在多台同样的服务器之前，加入一个“调度员”，将用户的访问请求尽量平均分配给它们，导致每台服务器的负担下降，因而出问题的几率也就下降了。整个系统的简单示意如图 5-12 所示。

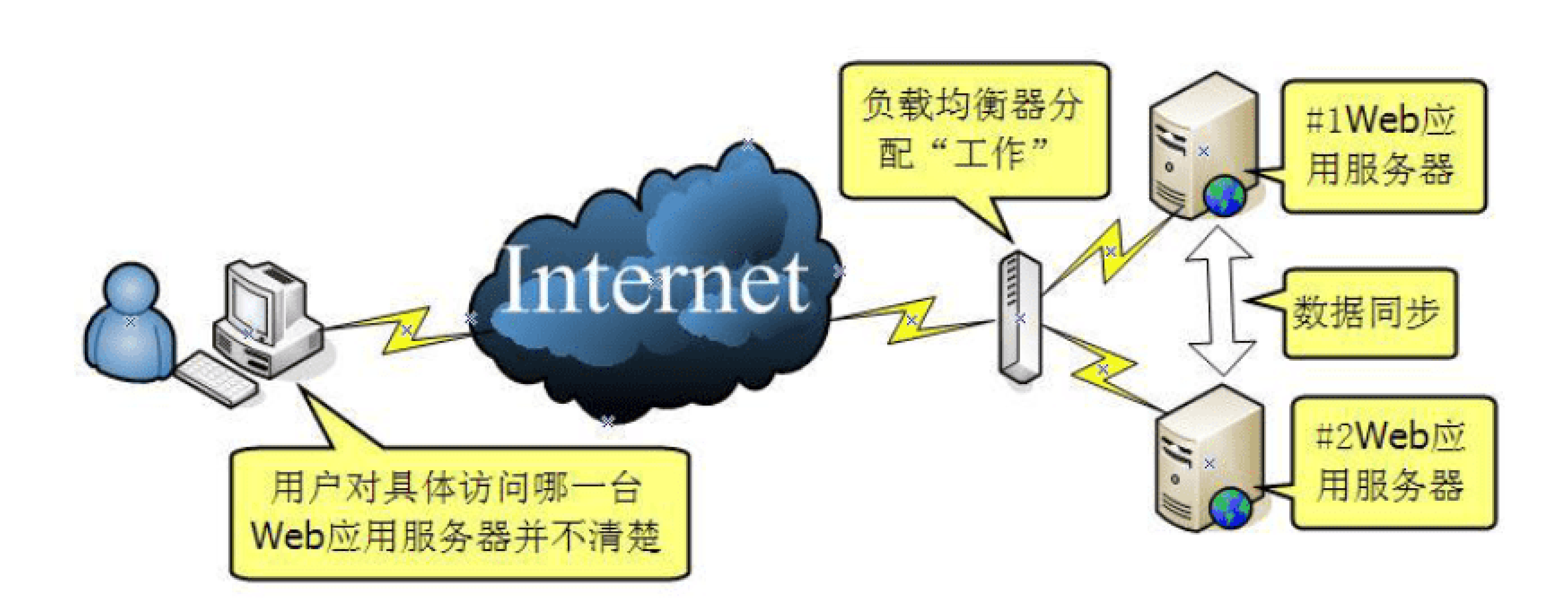


图 5-12　网站服务器的负载均衡

在实际工作中，还有其他一些方法，人们也往往综合使用这些方法来达到网站的 7×24（7 天，每天 24 小时，也就是全天候）可访问目标。

【失败恢复测试的意义】

失败恢复测试一般都是人工进行，有点类似奥运会开幕式的彩排。在模拟一台 Web 服务器出现故障的时候，验证另外的服务器能否接管用户的请求。当然，目前这些工作都已经可以由专用软件、硬件厂商来提供服务，只需要购买就可以了，实际的操作一般不会由测试工程师参与，技术经理和网络管理参与即可。但是，作为一个 Web 性能测试工程师，没有进行失败恢复测试的意识是不合适的。在以后的章节中，我们将和小白一起编写测试计划，在其中，应该给失败恢复测试留出一定的文字、安排一定的时间。

5.2　Web 性能测试方法的比较与共性

在 5.1 节中，我们学习了 9 种 Web 性能测试的方法，在本节中通过总结的形式，指出它们的不同，并从操作步骤等方面指出它们的共性，供读者参考。

5.2.1　各种 Web 性能测试方法的比较

性能测试包含的这 9 种具体测试方法，实际上是从不同的角度和出发点来考察 Web 应用的性能表现。在这些方法中，有的关注“点”，比如负载测试所关注的性能极限；有的关注“面”，比如性能测试所关注的一般性能情况；还有的关注“变化”，比如尖峰冲击测试所关注的用户数量突然增加。具体来说：

- ❑ 性能测试是整个广义的性能测试中最基本的、也是必备的方法，它获得网站总体性能的评估，使网站的各个参与者对情况有基本了解。它是偏向总体、宏观的。
- ❑ 负载测试通过不断给 Web 应用增加负荷的过程来获得系统能够承受的最大压力数值。它确定了性能的最大限度和范围。
- ❑ 尖峰冲击测试、并发测试都是从在线用户数量这个因素来考察 Web 应用是否健壮，不同的是尖峰冲击测试注重在线用户的变化率，并发测试注重在线用户的绝对值。
- ❑ 压力测试考察了 Web 应用在服务器较大负荷条件下的表现，与并发测试、耐久度测试（或可靠性测试）分别侧重在线用户数量与运行时间两个具体因素不同。
- ❑ 配置测试为现有 Web 应用发挥最大效能提供了一种途径。它更面向网站性能优化，而不是发现网站性能上的 Bug。
- ❑ 失败恢复测试则为网站把好最后一道关口，验证一旦网站出现问题后能够快速恢复的能力。

【各性能测试方法执行的时机】

正如了解一个人需要从多方面去获得信息一样，了解网站的真实性能情况也需要以上各种性能测试方法的配合。在实际工作中，各项测试的开始时间不一定是一成不变的，可以因公司规定、制定测试计划的工程师个人理解与习惯不同而灵活设置。笔者个人采取的性能测试顺序如图 5-13 所示。

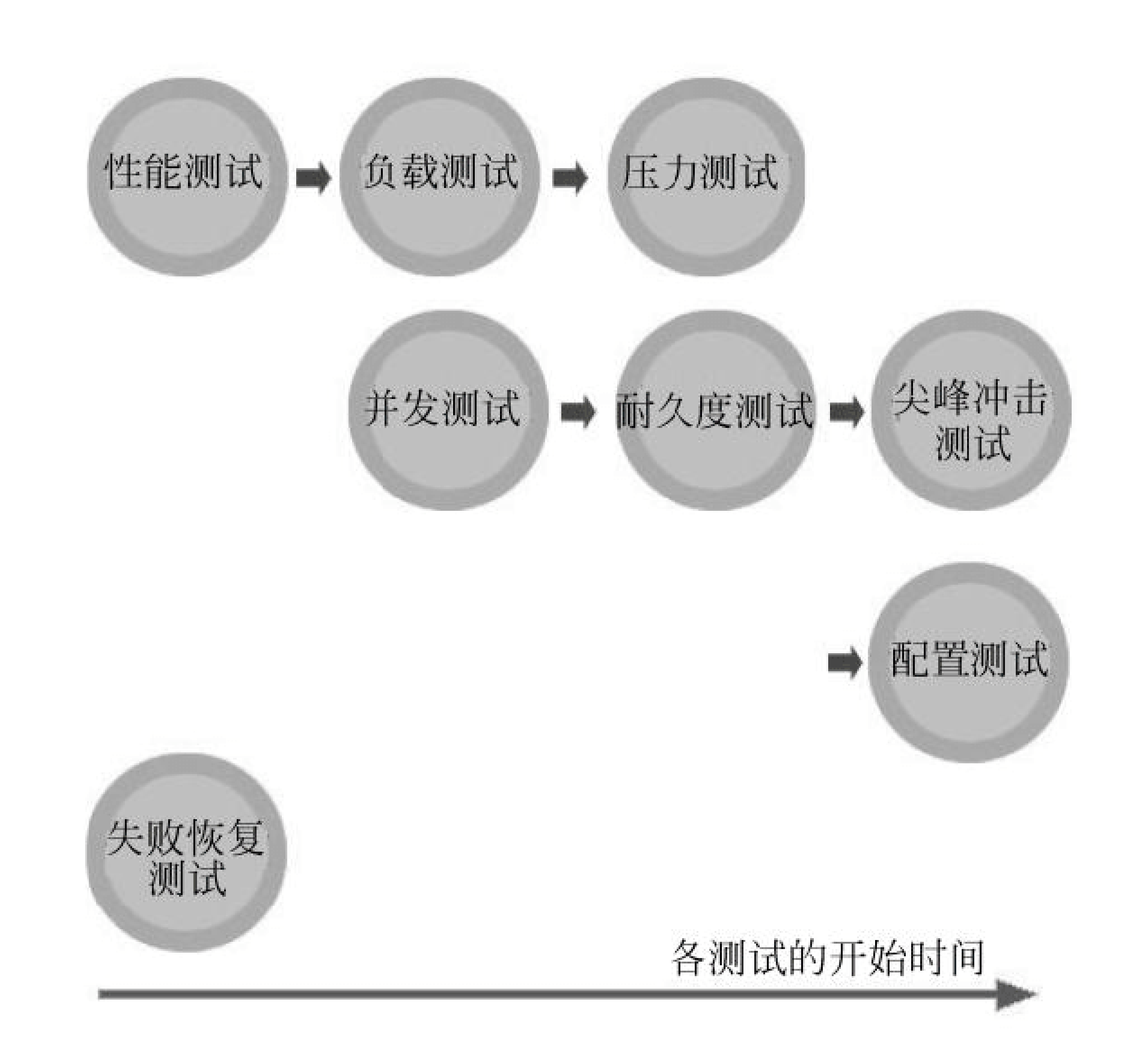


图 5-13　笔者采用的各性能测试方法开始时间顺序示意

5.2.2　各种 Web 性能测试方法的相同点

虽然各种 Web 性能测试在测试理念上有一些区别，但它们具有更多的相同点，这些相同点导致各种性能测试方法在实际工作中往往界限并不非常清楚，“你中有我，我中有你”：

（1）它们同属于性能测试的范畴，都遵循性能测试的 3 个目的。

- 获得 Web 应用的性能表现情况。
- 发现并验证、修改 Web 应用中影响性能的 Bug。
- 为网站性能优化提供数据参考。

（2）它们的测试步骤均类似，都具有如下阶段。

- 测试计划阶段：约定测试所采用的具体方法、时间、资源。
- 测试准备阶段：确定测试采用的 Web 应用版本号、准备测试环境。自动测试还需要准备、录制模拟场景等的脚本等工作。
- 测试实施阶段：进行测试，获得测试结果。
- 测试分析阶段：对结果进行分析，发现 Bug 或性能优化关键点。
- 测试报告发送：将分析结果进行总结，发送给网站相关人员。
- 测试总结阶段：综合各种测试结果，积累经验，为下一次测试打下坚实基础。

测试步骤在后面的章节还要具体涉及，比如测试计划的编写，测试结果的分析，测试报告的发送等，这里就不展开讨论了。

5.3　本 章 小 结

本章主要介绍了 Web 性能测试的目的和具体测试的分类/方法，这 9 种方法分别是：

- ❑ 性能测试（Performance Testing）；
- ❑ 压力测试（Stress Testing）；
- ❑ 负载测试（Load Testing）；
- ❑ 并发测试（Concurrency Testing）；
- ❑ 配置测试（Configuration Testing）；
- ❑ 耐久度测试（Endurance Testing）；
- ❑ 可靠性测试（Reliability Testing）；
- ❑ 尖峰冲击测试（Spike Testing）；
- ❑ 失败恢复测试（Failover Testing）。

其中，性能测试是整个广义性能测试中最基本、也是必备的一个方法，能够获得被测试网站的一般性能。负载测试、尖峰冲击测试主要是查看 Web 应用在极端情况下的表现，压力测试、并发测试和耐久度测试（可靠性测试）分别考察 Web 应用在较大负荷、多用户和长时间运行 3 种条件下的表现，验证了网站受到具体重要因素影响的程度。配置测试则是为现有 Web 应用的优化提供了一种途径。失败恢复测试则为提高网站的可用性、预防和补救网站出现的问题提供了验证的方法。

总之，合理、灵活地利用以上这些测试方法，可以使网站在开发阶段、测试阶段、部署阶段、运行阶段有关性能方面都有全方位的质量报告与质量保证，从而给予网站更优化的性能、用户更快速的响应，以及公司其他相关部门更多的信息、更大的信心。

第 6 章　性能测试计数器

小白在第 5 章开始着手的 3 项工作：熟悉性能测试方法、了解性能测试计数器和熟悉一种性能测试工具软件，已经有了一定基础。下一步，小白打算用几天的时间熟悉操作系统的软硬件各部分、Web 应用服务器以及数据库服务器的性能计数器。在本章，读者将和小白一起学习这方面的详细内容。

每一种操作系统为了开发、使用、管理者的方便，都会提供所谓的“性能计数器”（Performance Counter）来监测当前状态下系统的性能。对于不同的操作系统，这些计数器自然也不一样。那么，我们需要学习哪些系统的性能计数器呢？要回答这样的问题，首先需要对当今世界的主流操作系统份额有所了解。

【主流操作系统的市场份额】

我们知道，每台电脑一般都需要安装操作系统才能工作：有的是 Windows，有的是 Linux，还有的是各种各样的 Unix 等。操作系统负责管理计算机的软硬件。根据 Market Share 公司的统计，截至 2008 年 9 月，主流浏览器的市场份额如图 6-1 所示。

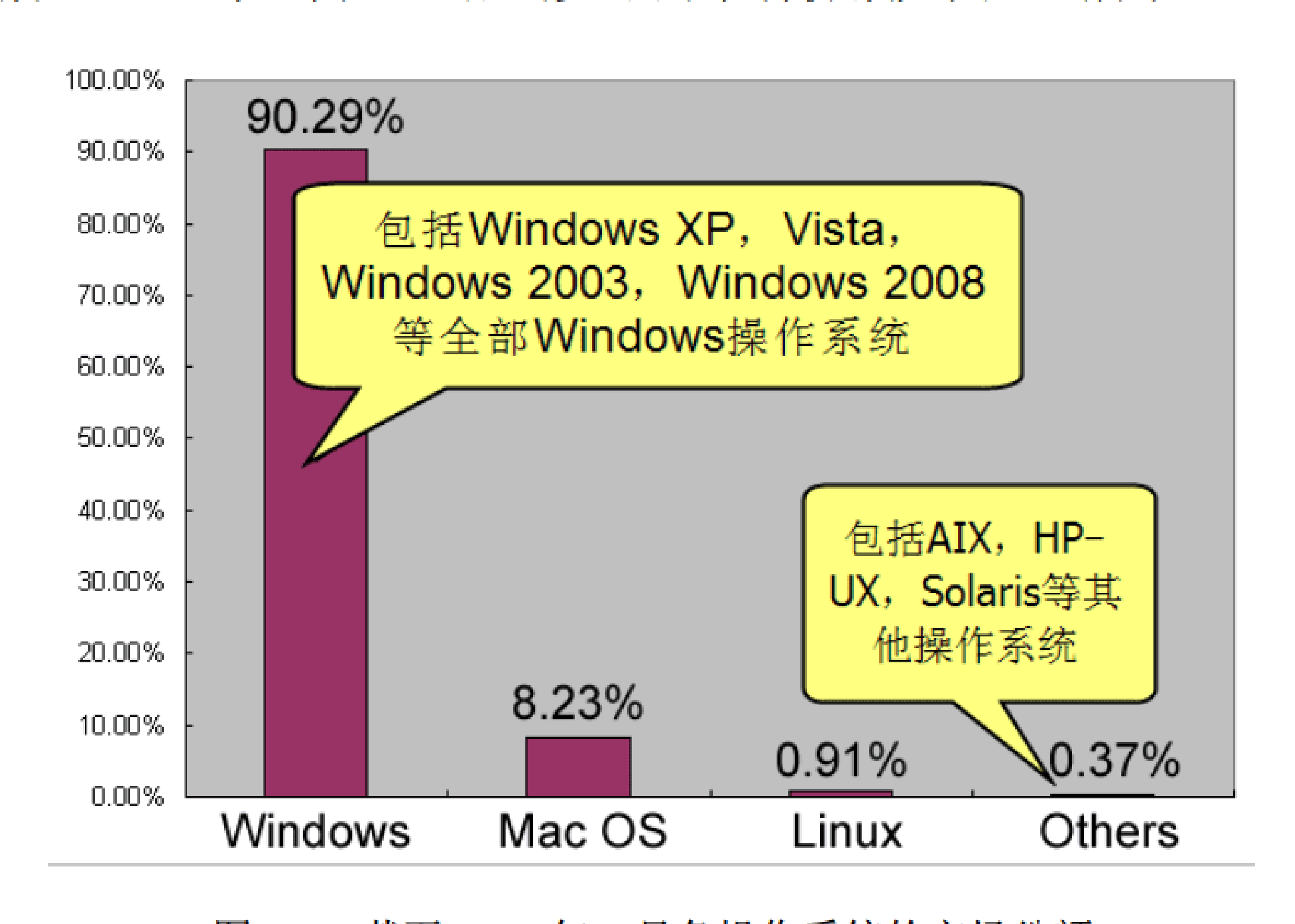


图 6-1　截至 2008 年 9 月各操作系统的市场份额

对于 Linux 和 Unix，不要被它们较低的市场占有率而迷惑：它们大部分都是安装于服务器端，而我们的 Web 应用正是运行在各类服务器上，Web 应用的性能测试也主要在服务器端进行，所以，了解它们的性能指标至少与 Windows 下的性能指标同等重要。

排名前 3 位的操作系统，都有各自的代表浏览器，便于用户访问网站。它们分别是 IE、Safari 和 Firefox，也恰恰是浏览器市场占有率排名的前 3 位（虽然次序有所不同）。

对于一个合格的性能测试工程师来说，我们有必要熟悉 Windows、Mac 和 Linux 这 3 种系统下有哪些性能计数器，以及查看它们的方法。

6.1　性能计数器简介

性能计数器（Performance Counter，或者简称为 Perf Counter）是由某个软件提供，能够显示该软件系统当前运行状况的一些指标。它好像汽车仪表盘上的速度计、油量计和转速表，我们可以通过性能计数器的数值来评估当前软件的负荷情况，更可通过它们来进行性能测试结果的分析。因此，性能计数器既和性能测试的具体方法有关，也和性能测试的结果分析有关，有必要放置于性能测试具体操作章节之前介绍。

和 Web 应用相关的软件，比如操作系统、应用服务器（Application Server）和数据库，都提供了性能计数器。

值得注意的是，性能计数器所反映的都是软件对系统资源的某个方面、某个部分的占用情况。因此，单一的一个性能计数器并不能全面地反映系统状态。日后我们在进行性能测试结果分析的时候，一般都要对多个性能计数器进行综合的分析。在本章我们将依据操作系统、应用服务器和数据库的顺序，对它们提供的性能计数器做一一讲解，读者要认识到它们不是孤立的，在实际工作中要注意综合使用。

在 3 种 Web 应用所依赖的软件中，操作系统无疑是最重要的。操作系统也是一种软件，而且它作为基础平台，支撑应用服务器、数据库和 Web 应用。操作系统提供的性能计数器用来监控整个操作系统的性能表现。根据本章开头所列出的主流操作系统，下文将按照 Windows、Mac 和 Linux 这 3 种操作系统的性能计数器。对于 Unix 系统，由于与 Linux 或者 Mac 系统类似，读者可以在具体工作中遇到不一致的地方查看系统手册获取进一步的信息，这里就不赘述了。

6.2　Windows 系统下的性能计数器

由于目前市场占有率最大的操作系统是各种 Windows，它们对于读者来说也容易接触到，我们就从它开始。

6.2.1　Windows 系统下性能计数器数值的直观获得

性能计数器是操作系统提供的指标，因此我们需要有一些界面或者命令来获得具体的数值。Windows 系统提供了一些图形界面程序完成这样的工作。

如果需要获得系统总体的性能，可以通过 Windows 自带的任务管理器：在系统任务栏中右击，在弹出的快捷菜单中选择“任务管理器”命令，弹出界面如图 6-2 所示。

通过在该界面上查看进程、性能等标签，我们可以获得当前系统中进程占用 CPU 的时间（如图 6-2 中的进程列表）、内存数量、缓存数量、网络流量、登录用户等多种信息，对性能有基本的了解。图 6-3 用类似示波器或者心电图的形式显示了当前 CPU 和页面文件的使用情况，看起来非常专业。

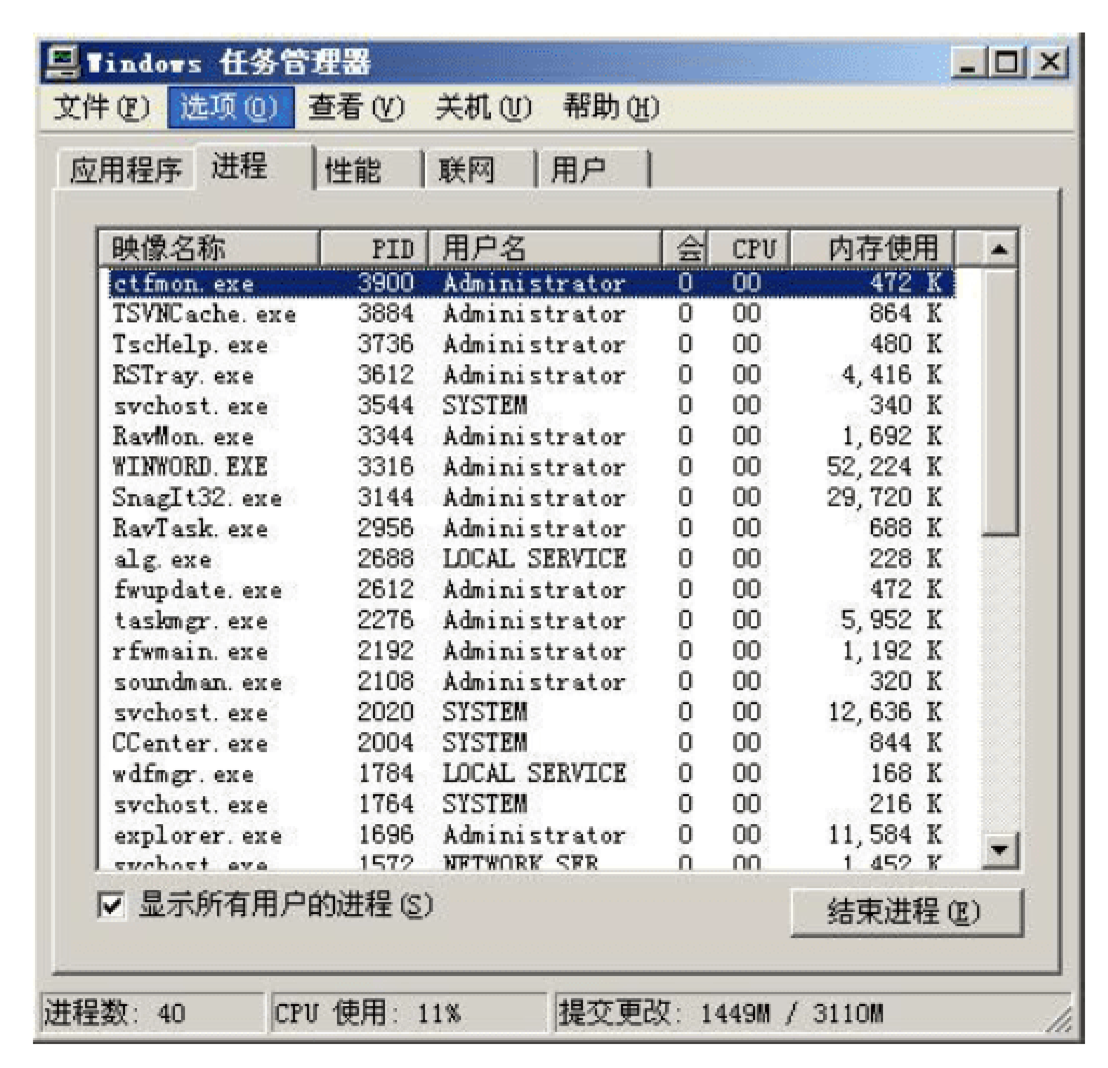


图 6-2　Windows 任务管理器的进程标签

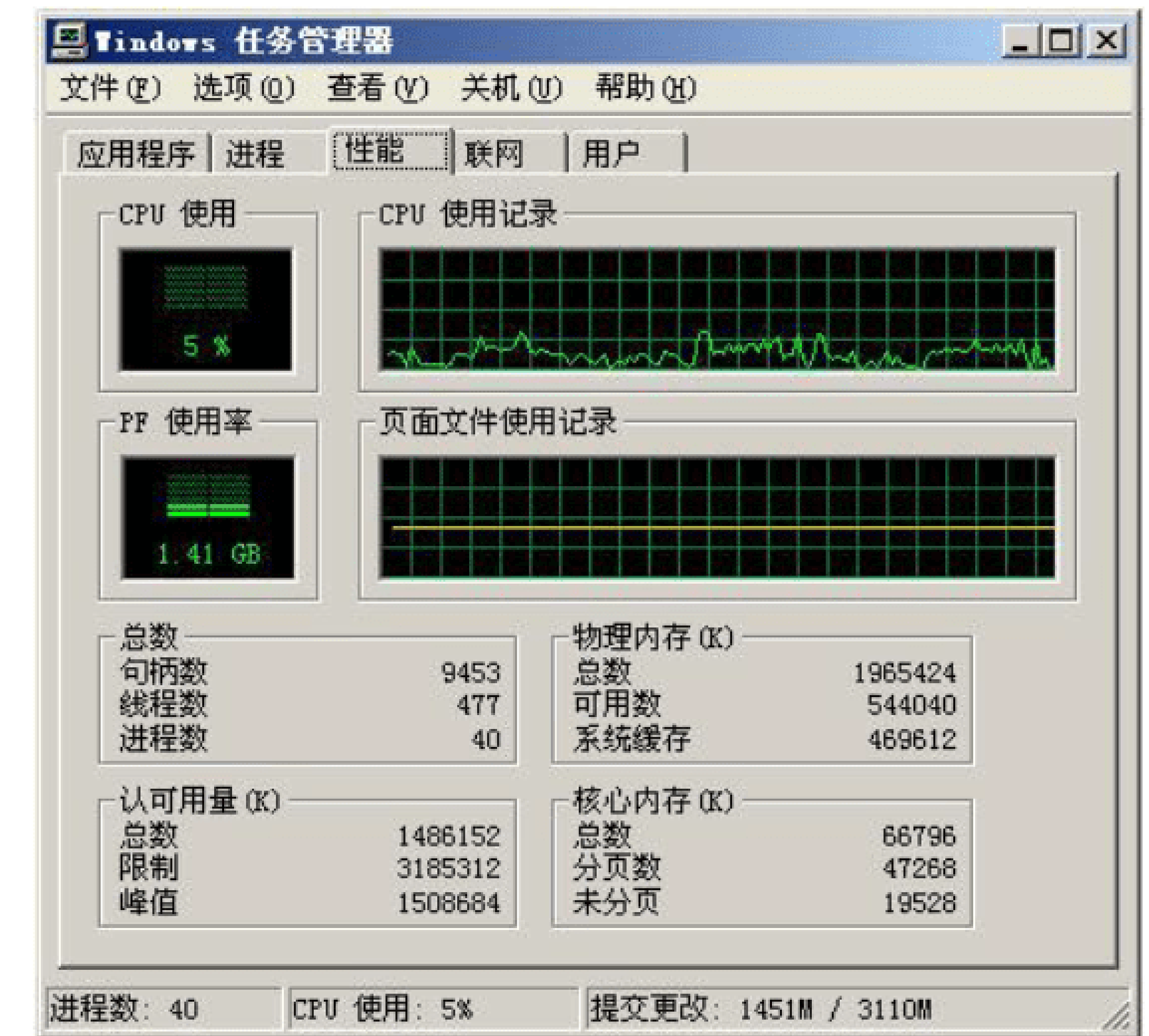


图 6-3　Windows 任务管理器的性能标签

但是，光了解以上这些信息是不够具体的，准确反映性能需要更多更细致的性能计数器。实际上，性能计数器在 Windows 系统中是用“性能监视器”（Performance Monitor）这个程序来打开的。

6.2.2　Windows 系统下性能监视器的使用

在 Windows 系统中依次单击“开始”|“运行”命令，在弹出的“运行”对话框中输入 perfmon 并回车，就能够打开系统自带的性能程序，其界面如图 6-4 所示。

在图 6-4 中，默认列出了 3 个计数器，在图的右下栏。它们分别代表了内存、物理硬盘和处理器的某一个性能指标。图右边的大部分是这几个被选择计数器的变化曲线，以一根红线为界，红线向右移动表示时间的推移。

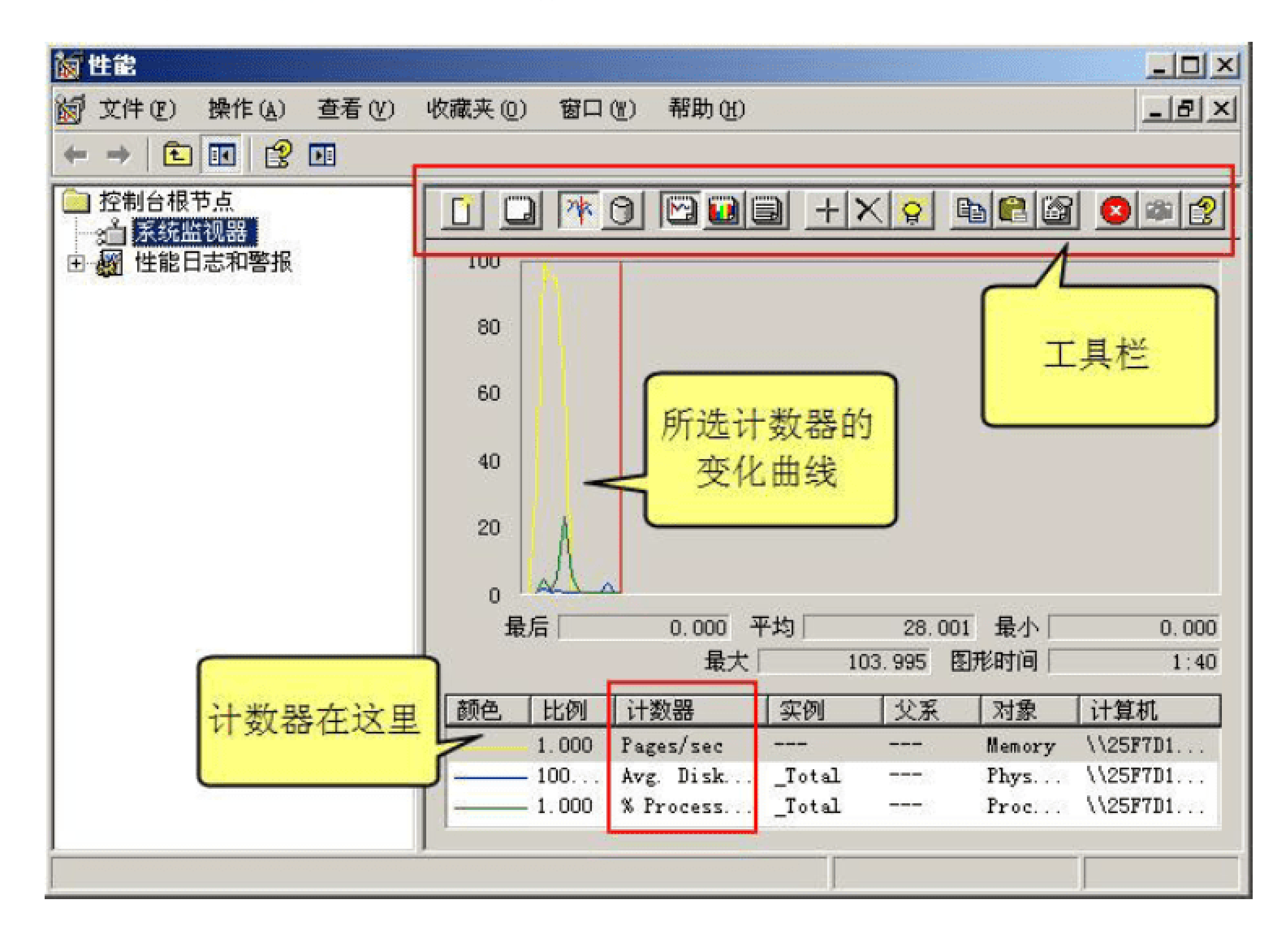


图 6-4　Windows 系统下的性能界面

1．增加、删除计数器

在图 6-4 中变化曲线的上方是一个拥有很多图标按钮的工具栏，其中和增加、删除计数器功能有关的按钮在图 6-5 中用方框标出。单击加号按钮增加计数器，弹出对话框如图 6-6 所示。

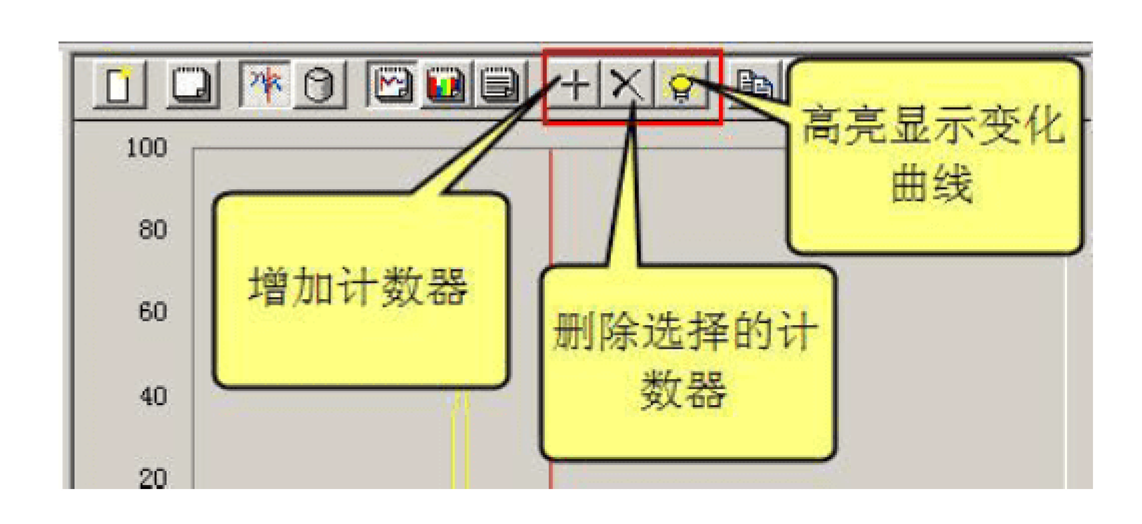


图 6-5　工具栏上增加与删除计数器的按钮

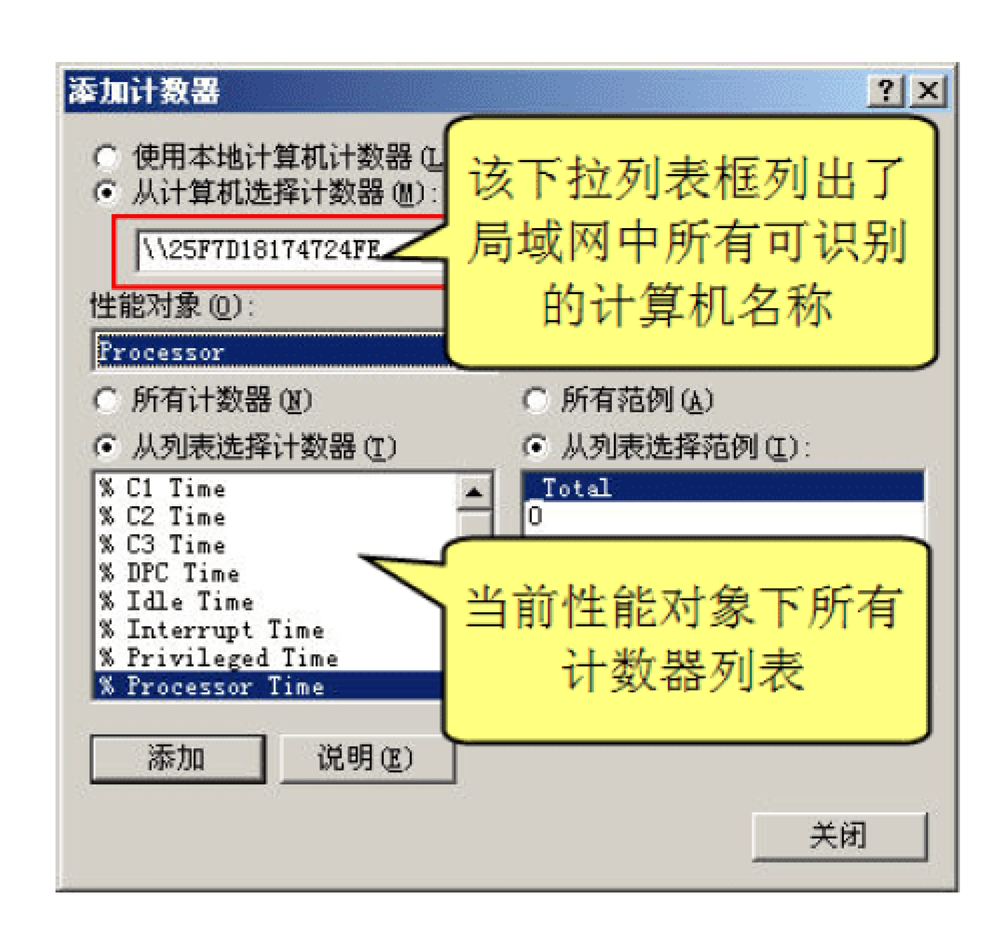


图 6-6　添加计数器的界面

在图 6-6 中有几点需要特别指出：

- 可以通过在图 6-6 中单击“从计算机选择计数器”单选按钮，并在其下方下拉列表框中选择某一项（每项对应局域网中的一台电脑），即可实现在本地监控该电脑性能计数器的功能，这对于同时监控多台服务器的性能是非常有用的。
- 性能对象实际上是各个子系统，比如内存、处理器、磁盘等都是一个性能对象。每一个性能对象都包含多个计数器，这样归类便于管理和查看。性能对象不限于硬件，软件同样可以是性能对象，比如.Net CLR（第 2 章介绍过的.Net 平台的基础组成部分）就有多个性能对象对应。
- 计数器前若带有百分比，一般表示占用 CPU 总工作时间的百分比，越大说明越耗费处理器资源。
- 如果不确定英文字母代表的计数器的具体含义，可以选择该计数器，单击下方的说明按钮，会弹出该计数器的详细说明。

在计数器选择完毕后，单击“添加”按钮，图 6-4 右下方的列表将增加这个计数器。对于计数器的删除操作，只需要在图 6-4 右下方的计数器列表中单击待删除的项，然后在工具栏中单击由叉号代表的删除按钮即可。

目前普遍的服务器都拥有多个同类型的硬件对象，这样它们会分别有各自的性能对象和相应的计数器。比如：

- 如果服务器系统内有多个 CPU，则“处理器（Processor）”对象类型将拥有多个实例，性能对象的列表框中将分别列出。
- 如果系统有两个磁盘，则“物理磁盘（PhysicalDisk）”对象类型有两个实例。

但是，有些对象类型，比如“内存 (Memory)”，无论内存条有几根，则只有一个实例。对于和“服务器（Server）”等性能对象，也是如此。

如果对象类型有多个实例，则可以针对每个实例将计数器添加到跟踪统计中。当然在更多的情况下，人们会一次性针对所有实例将计数器添加到统计中。

如果觉得默认的计数器变化曲线对于当前的数值来说显示方式不是很合适，也可以通过单击工具栏上的显示控制按钮进行切换，如图 6-7 所示。可以选择的显示方式有 3 种：

- 计数器变化曲线方式。这也是默认的显示方式。
- 计数器数值直方图方式。
- 计数器数值报表方式。



图 6-7　切换计数器数值显示方式

2. 创建计数器集与清除显示

实际测试工作中，有时候需要针对不同的测试目的和场景，关注不同的计数器，这时界面上已有很多的计数器，逐个地进行删除很麻烦，因此可以通过创建新的计数器集，把现有的计数器都清空，重新添加。新计数器集按钮在工具栏上的位置如图 6-8 所示。如果需要在新创建的计数器集中不必通过复杂的界面进行计数器的选择，可以通过复制、粘贴按钮直接添加计数器。由于这些按钮都是标准的 Windows 图标，就在图中列出了。

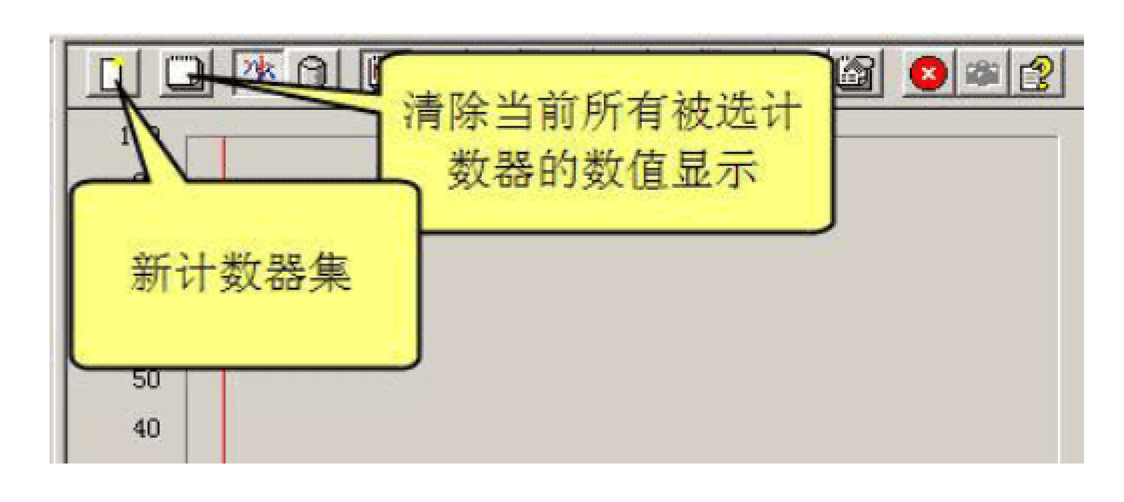


图 6-8　新计数器集和清除计数器数值显示

另外，如果打算在图 6-4 中清除当前各计数器的数值结果，可单击工具栏中的第 2 个按钮，清除即可。

3. 属性设置和冻结显示

如果需要对计数器的显示结果进行自定义的设置，比如背景、曲线的颜色等，可单击

工具栏上的属性按钮。如果需要暂时中止时间线的推移，在当前点上进行一番分析，单击冻结显示按钮即可，非常方便，如图 6-9 所示。

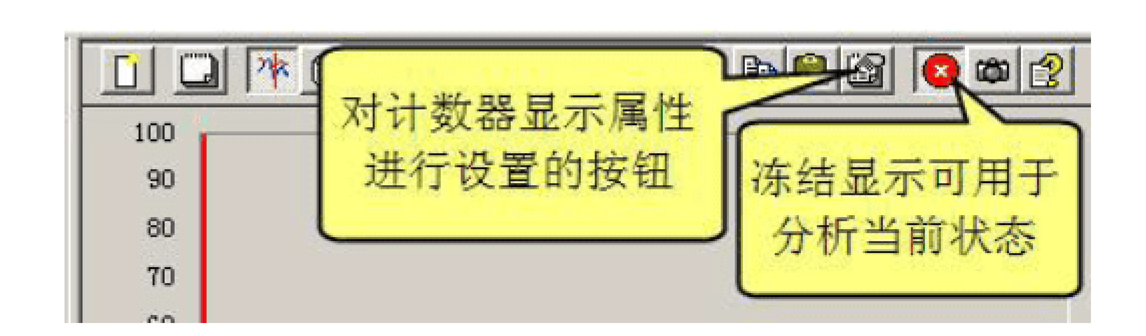


图 6-9　属性按钮和冻结显示按钮

4．性能日志的建立

前面介绍的各种操作对于实时查看计数器的值是很有用的，但是如果需要长时间的获取这些信息，特别是需要晚上下班时间内的数据，总不能常年蹲守在电脑前面查看吧，而且靠手工也无法记录准确和全面。因此，将计数器的变化结果自动记录成为文件，等到我们上班后再进行分析是一个很好的解决办法。

性能程序也提供了这样的界面，如图 6-10 所示，可以通过增加日志来保存一段时间内关注的计数器数值的变化。在图 6-10 的计数器日志空白处右击，在弹出的快捷菜单中选择“新建日志设置”选项。在弹出窗体的“当前日志文件名”文本框中可输入自定义的日志文件存放位置，还可以自定义被关注的性能计数器对象，如图 6-11 所示。单击“确定”按钮使得设置生效。

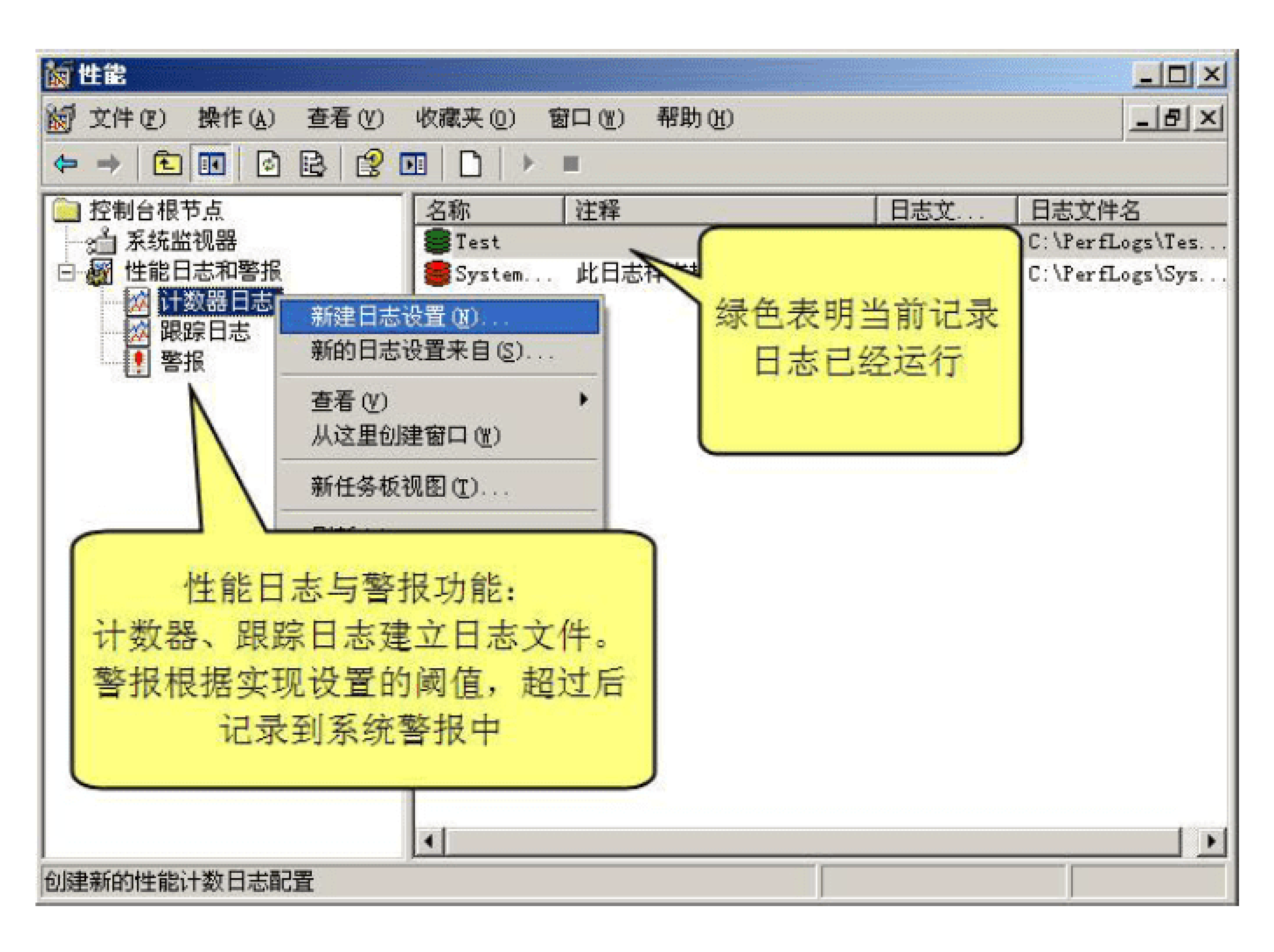


图 6-10　建立日志文件供分析和查看

值得注意的是，在图 6-10 中，还有关于警报的设置。实际上，警报和日志类似，也是对一段时间内计数器数值变化的监控，但是，它会要求使用者提前设置一个阈值，比如 CPU 占用率 75%，如图 6-12 所示。如果 CPU 占用率这个计数器真的超过该设定数值，会在系统的事件查看器中增加一个信息记录。

读者可以通过在系统任务栏中依次单击“开始”|“运行”命令，然后在“运行”文本框中输入 eventvwr 之后回车，就可以打开事件查看器（Event Viewer）程序。在其左边“事件查看器”列表中选择“应用程序”项目，可以查看各应用程序向系统发出的日志信息，

如图 6-13 所示。

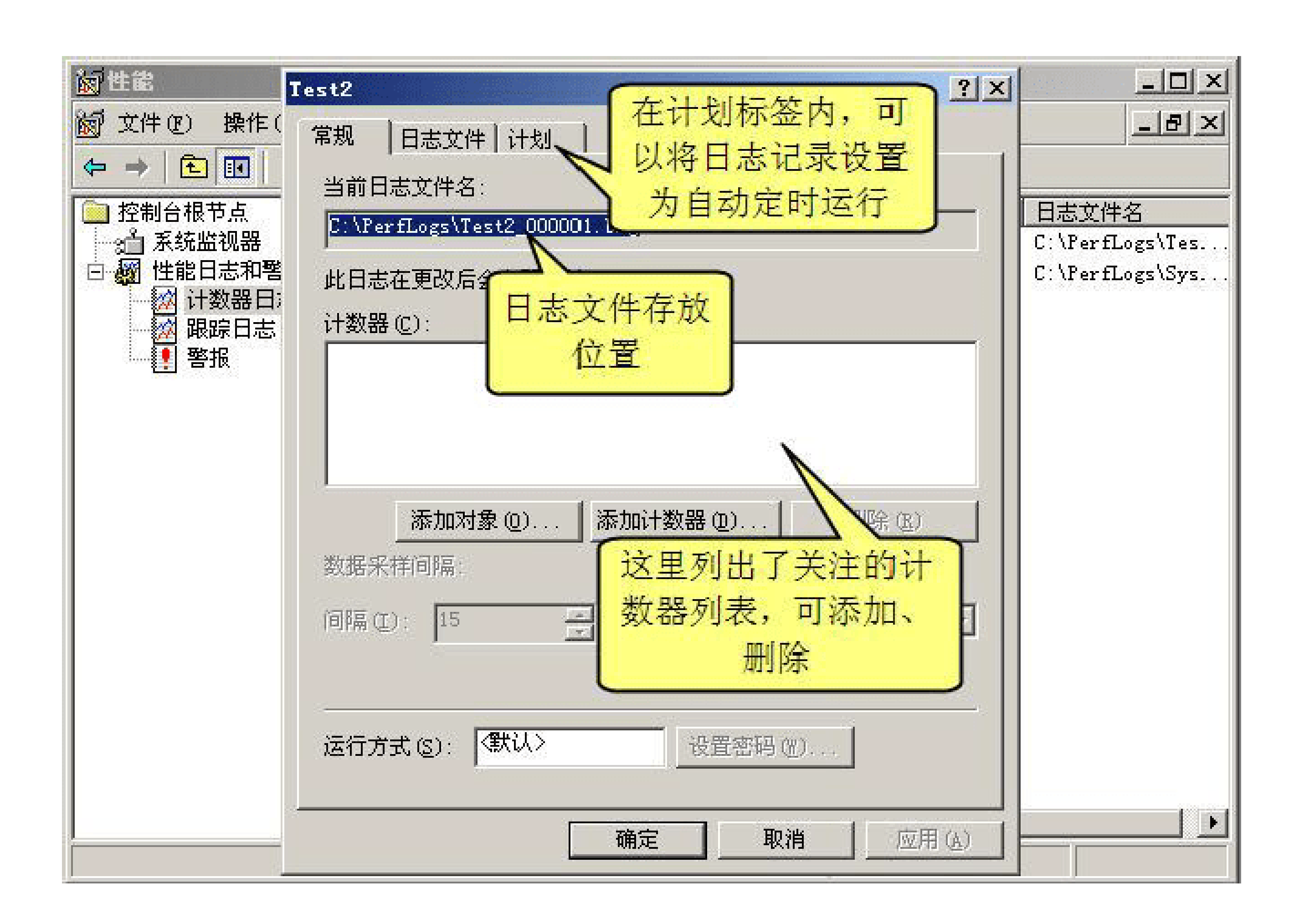


图 6-11　计数器日志文件的设置

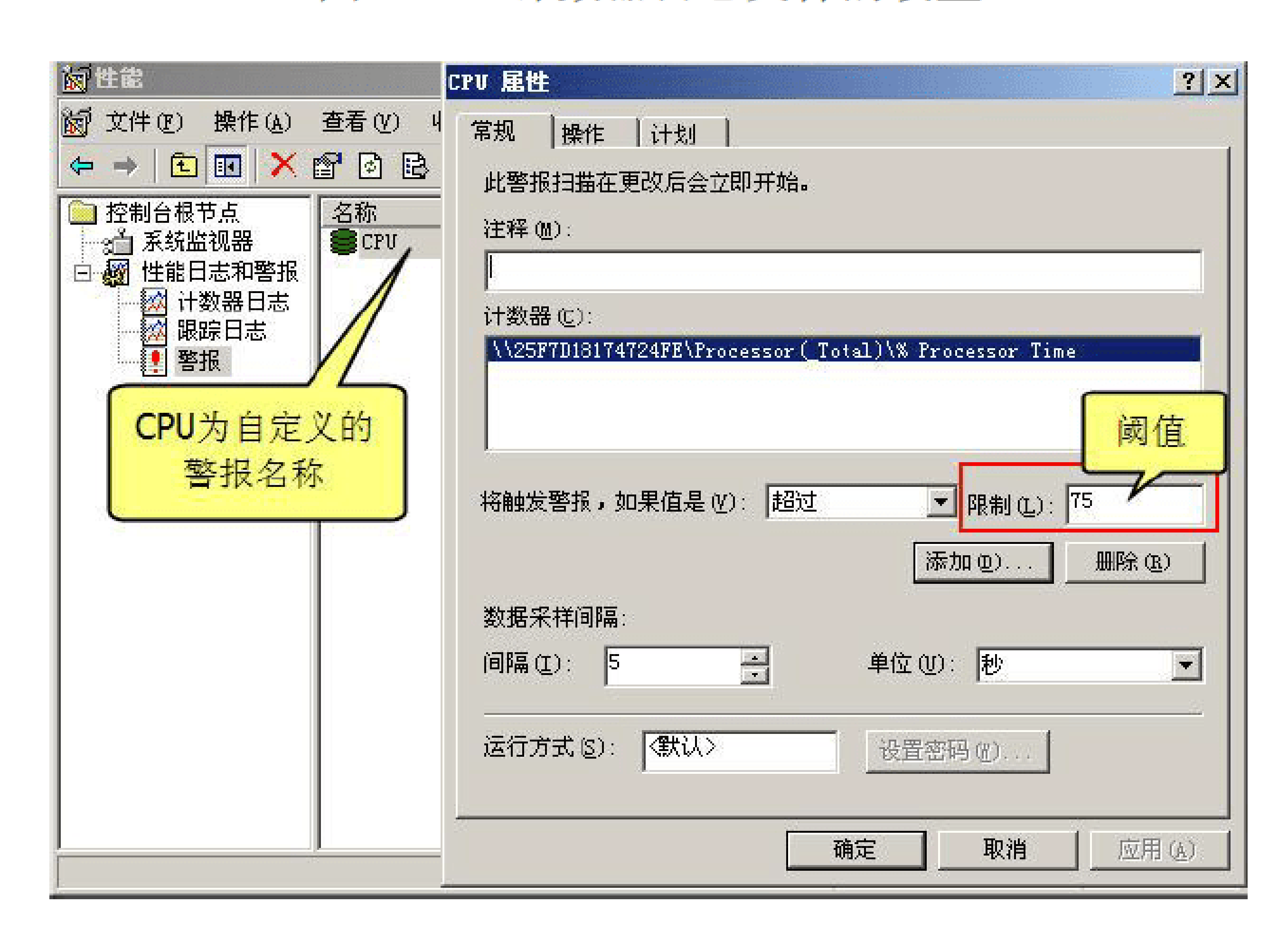


图 6-12　警报的创建与阈值

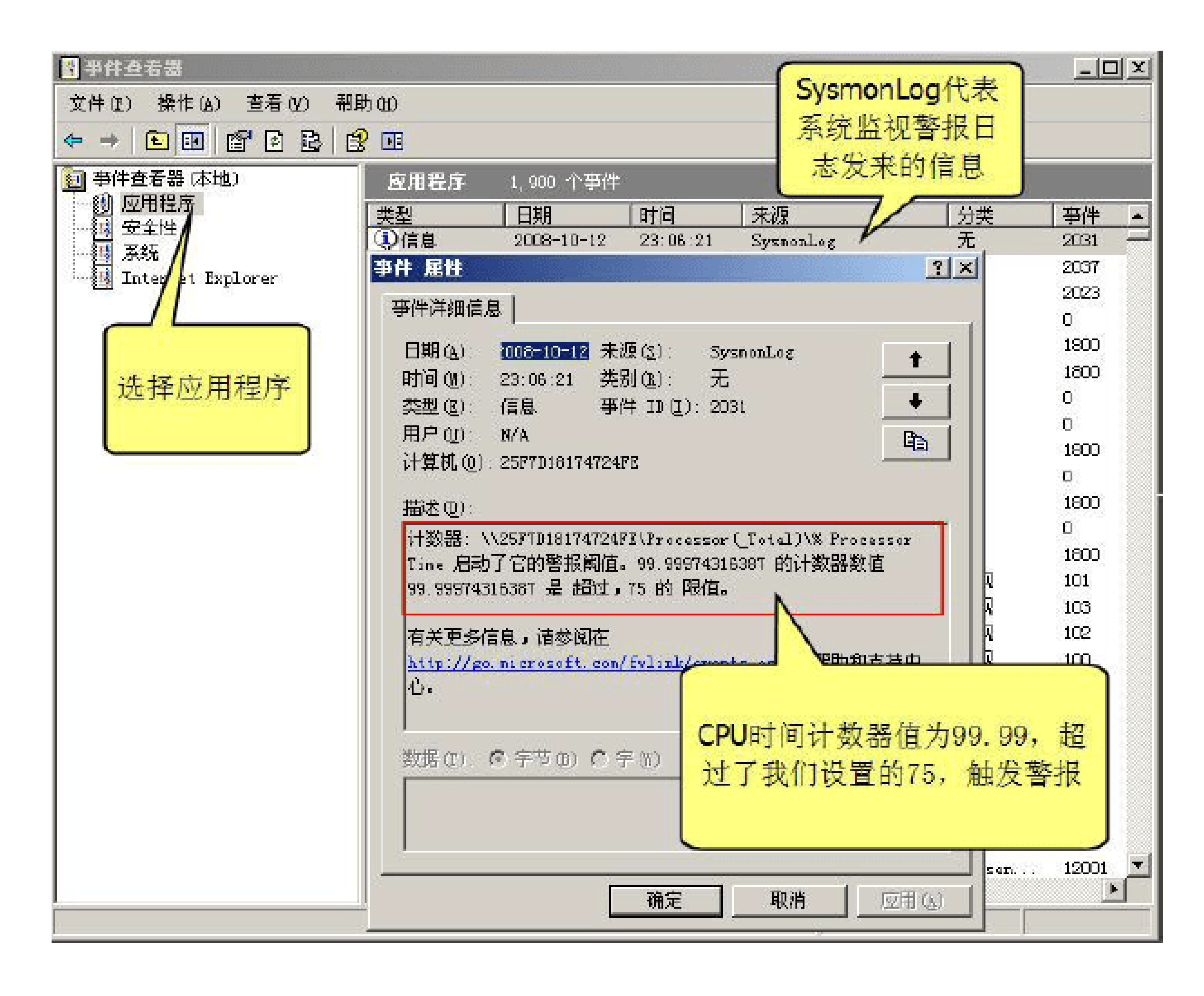


图 6-13　Windows 系统下的事件查看器可以查看性能警报

5．typePerf命令自动获取性能日志

除了上述方法，读者还可以在命令行中输入 typeperf 命令来自动获取某个或者多个性能计数器的实时信息。typePerf 命令的语法如下：

```
typeperf "\Processor(_Total)\% Processor Time" -o c:\processor.log
```

该命令将每秒取一次本机处理器时间计数器的数值，将其记录在 C 盘下的 processor.log 文件中。

有了 typeperf 命令，可以通过编程或者批处理的方式实现获取性能计数器的自动化。除此之外，还可以调用系统提供的 API 接口，详见 6.2.3 节。

性能日志很有用处，我们不可能在有限的篇幅中完全讲解所有的知识点，好在性能程序是操作系统自带的，从 Windows XP 到 Windows Server 2008 都有，使用比较简单，读者可以在工作中进一步探索。

6.2.3　编程获得 Windows 系统下性能计数器的方法

实际上，还有其他的一些方法可以用于获得 Windows 系统下的性能计数器，比如系统管理员可以利用 WMI（Windows Management Interface）脚本、开发工程师可以利用.NET 平台提供的某些类库功能来实现。这些方法特别适用于创建企业内部使用的自动性能测试工具。代码 6-1 显示了利用 VBScript 查询 WMI 接口获得当前硬盘性能计数器等信息的方法。

代码 6-1　通过 VBScript 与 WMI 查询硬盘性能计数器信息

```
On Error Resume Next
strComputer = "." '表示本机
Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\cimv2")
'获取 WMI 对象
Set colItems = objWMIService.ExecQuery("Select * from Win32 PerfRawData
PerfDisk PhysicalDisk",,48) '查询 WMI 接口
For Each objItem in colItems '下列每一行均为显示一个性能计数器的值
    Wscript.Echo "AvgDiskBytesPerRead: " & objItem.AvgDiskBytesPerRead
    Wscript.Echo "AvgDiskBytesPerRead Base: " & objItem.AvgDiskBytesPer
    Read Base
    Wscript.Echo "AvgDiskBytesPerTransfer: " & objItem.AvgDiskBytesPer
    Transfer
    Wscript.Echo "AvgDiskBytesPerTransfer Base: " & objItem.AvgDisk
    BytesPerTransfer Base
    Wscript.Echo "AvgDiskBytesPerWrite: " & objItem.AvgDiskBytesPerWrite
    Wscript.Echo "AvgDiskBytesPerWrite Base: " & objItem.AvgDiskBytes
    PerWrite Base
    Wscript.Echo "AvgDiskQueueLength: " & objItem.AvgDiskQueueLength
    Wscript.Echo "AvgDiskReadQueueLength: " & objItem.AvgDiskReadQueue
    Length
    Wscript.Echo "AvgDisksecPerRead: " & objItem.AvgDisksecPerRead
    Wscript.Echo "AvgDisksecPerRead Base: " & objItem.AvgDisksecPer
    Read Base
    Wscript.Echo "AvgDisksecPerTransfer: " & objItem.AvgDisksecPerTransfer
    Wscript.Echo "AvgDisksecPerTransfer Base: " & objItem.AvgDisksecPer
    Transfer Base
    Wscript.Echo "AvgDisksecPerWrite: " & objItem.AvgDisksecPerWrite
```

```
    Wscript.Echo "AvgDisksecPerWrite Base: " & objItem.AvgDisksecPer
    Write Base
    Wscript.Echo "AvgDiskWriteQueueLength: " & objItem.AvgDiskWriteQueue
    Length
    Wscript.Echo "Caption: " & objItem.Caption
    Wscript.Echo "CurrentDiskQueueLength: " & objItem.CurrentDiskQueue
    Length
    Wscript.Echo "Description: " & objItem.Description
    Wscript.Echo "DiskBytesPersec: " & objItem.DiskBytesPersec
    Wscript.Echo "DiskReadBytesPersec: " & objItem.DiskReadBytesPersec
    Wscript.Echo "DiskReadsPersec: " & objItem.DiskReadsPersec
    Wscript.Echo "DiskTransfersPersec: " & objItem.DiskTransfersPersec
    Wscript.Echo "DiskWriteBytesPersec: " & objItem.DiskWriteBytesPersec
    Wscript.Echo "DiskWritesPersec: " & objItem.DiskWritesPersec
    Wscript.Echo "Frequency Object: " & objItem.Frequency Object
    Wscript.Echo "Frequency PerfTime: " & objItem.Frequency PerfTime
    Wscript.Echo "Frequency Sys100NS: " & objItem.Frequency Sys100NS
    Wscript.Echo "Name: " & objItem.Name
    Wscript.Echo "PercentDiskReadTime: " & objItem.PercentDiskReadTime
    Wscript.Echo "PercentDiskReadTime Base: " & objItem.PercentDiskRead
    Time Base
    Wscript.Echo "PercentDiskTime: " & objItem.PercentDiskTime
    Wscript.Echo "PercentDiskTime Base: " & objItem.PercentDiskTime Base
    Wscript.Echo "PercentDiskWriteTime: " & objItem.PercentDiskWriteTime
    Wscript.Echo "PercentDiskWriteTime Base: " & objItem.PercentDiskWrite
    Time Base
    Wscript.Echo "PercentIdleTime: " & objItem.PercentIdleTime
    Wscript.Echo "PercentIdleTime Base: " & objItem.PercentIdleTime Base
    Wscript.Echo "SplitIOPerSec: " & objItem.SplitIOPerSec
    Wscript.Echo "Timestamp Object: " & objItem.Timestamp Object
    Wscript.Echo "Timestamp PerfTime: " & objItem.Timestamp PerfTime
    Wscript.Echo "Timestamp Sys100NS: " & objItem.Timestamp Sys100NS
Next
```

该 VBScript 保存为 GetDiskPerf.vbs 文件，它可以通过如下方式运行：

单击 Windows 系统任务栏中的“开始”|“运行”命令，在弹出的“运行”对话框中输入 cmd 回车，系统将弹出命令行窗口。在其中输入 cscript getdiskperf.vbs 再回车，即可看到当前磁盘性能计数器的结果，如图 6-14 所示。

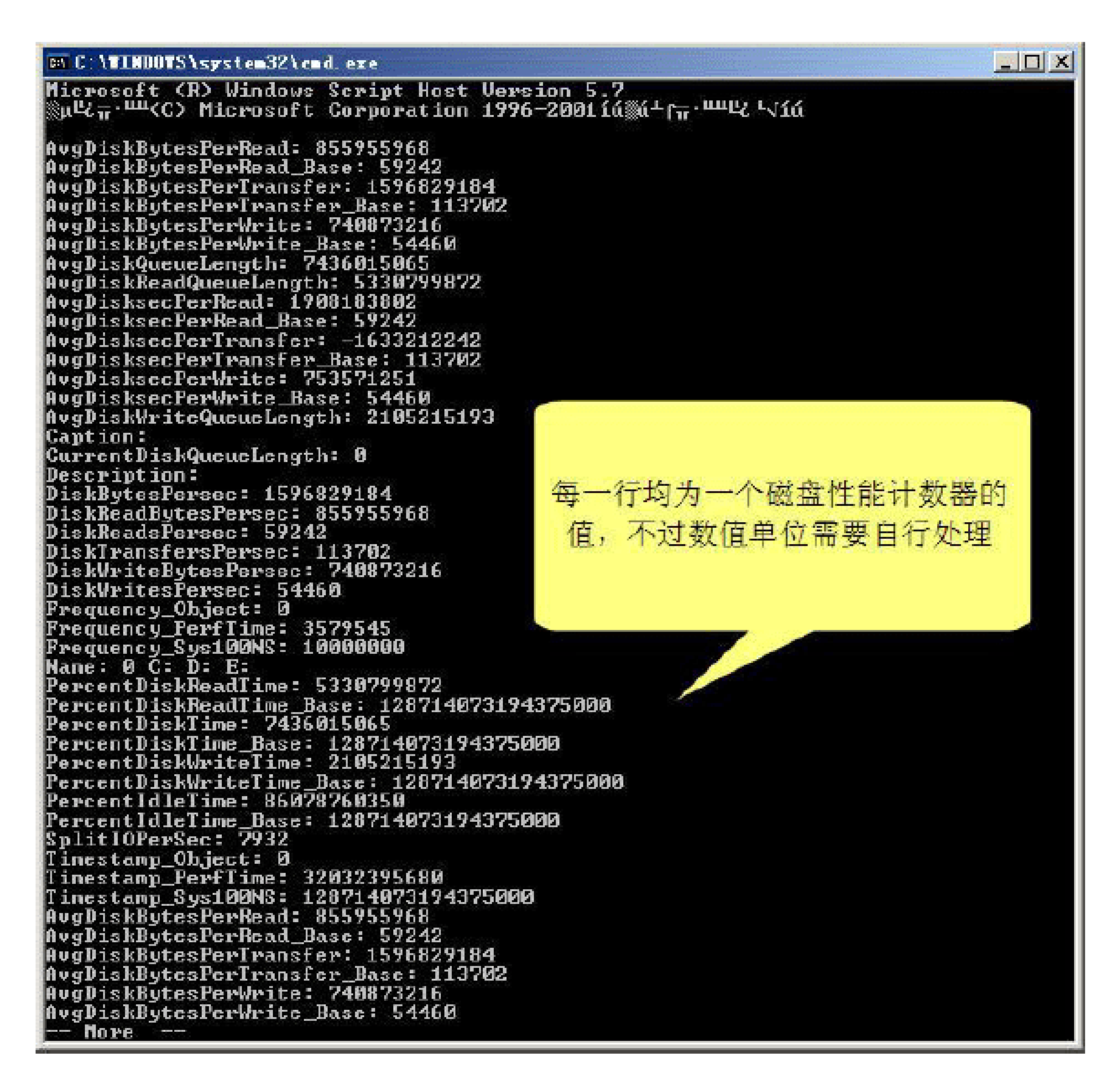


图 6-14　代码 6-1 的运行结果

利用其他的方法，比如在.NET 平台中调用 System.Diagnostics 命名空间中的 PerformanceCounterCategory 类，也可以查询到系统的性能计数器数值。有了以上这些编程接口，读者完全可以在实际工作中开发出更加符合实际用途的自动性能测试工具。

6.2.4　Windows 系统下常见的性能计数器的含义

在前面几节，我们学习了如何在 Windows 系统下获得当前各种计数器的数值。学习计数器的目的绝不是为了单纯地记录数值，而是从中分析出性能的缺陷和瓶颈。为了提高效率，快速发现问题，并不是所有的计数器都需要被关注。为简单明了起见，本节通过表格的形式列出了 Windows 系统下常见性能计数器的含义，关于内存、进程和处理器计数器的含义分别如表 6-1、表 6-2 和表 6-3 所示。

表 6-1　Windows系统下Memory（内存）性能对象中的几个重要计数器

计数器名称	计数器简要介绍
Available Mbytes Available Kbytes Available bytes	当前系统可用物理内存数量 （3 个计数器分别用不同的单位表示）
Page/sec	指每秒系统为解决硬页面错误（后面小节会介绍）而从磁盘读取、写入磁盘的页面数目
Page Faults/sec	为每秒出错页面的平均数量。由于在每个错误操作中只有 1 个页面出错，因此这也可以被认为是页面错误操作的数量。它包括硬页面错误与软页面错误（同样在后文讲述）。硬页面错误较多可以导致明显的系统性能变差
Page Reads/sec	它代表每秒钟读磁盘以解决硬页面错误的页面数量。比较计数器\\Pages Reads/sec 与另一个计数器\\Pages Input/sec 的数值，能决定每个操作读取的平均页面数量
Free System Page Table Entries	指系统没有使用的页表项目。这个计数值仅显示上一次的值，而不是一个平均值

表 6-2　Windows系统下Process（进程）性能对象中的几个重要计数器

计数器名称	计数器简要介绍
%Processor Time	是所有进程、线程使用处理器执行指令所花时间的百分比。指令是计算机执行的基础单位。线程是执行指令的对象，进程是程序运行时创建的对象
Page Faults/sec	指在当前进程中执行线程造成的页面错误出现的速度。当线程引用了不在主内存工作集中的虚拟内存页面，即会出现页面错误
Handle Count	由当前进程目前打开的句柄总数。数值等于进程每个线程当前打开的句柄的总数
Working set	即工作集。它反映了每个进程使用的内存页面的数量。如果服务器有足够多的空闲内存，页面就会被留在工作集中，当自由内存少于一个特定的阈值时，页就会被分页（Paging）清除出工作集至磁盘等地
Private Bytes	此进程所分配的无法与其他进程共享的当前字节数量。如果系统性能随着时间而降低，则此计数器可以说是内存泄漏的最佳指示器
Priority Base	进程的目前基本优先权。在一个进程中的线程可以根据进程的基本优先权提高或降低自己的基本优先权

表 6-3　Windows系统下Processor（CPU）性能对象中的几个重要计数器

计数器名称	计数器简要介绍
%Processor Time	指处理器用来执行非闲置线程时间的百分比。 它的计算方法是，测量范例间隔内非闲置线程活动的时间，用范例间隔减去该值。（每台处理器有一个闲置线程（System Idle），该线程在没有其他线程可以运行时消耗周期）。这个计数器是处理器活动的主要说明器，显示在范例间隔时所观察的繁忙时间平均百分比。这个值是用 100% 减去该服务不活动的时间计算出来的
%Users Time	用户模式下操作所耗费的 CPU 时间 用户模式与核心模式在后面介绍
%Privileged Time	核心模式下操作所耗费的 CPU 时间
Working set	即工作集。它反映了每个进程使用的内存页面的数量。如果服务器有足够多的空闲内存，页面就会被留在工作集中，当自由内存少于一个特定的阈值时，页就会被分页（Paging）清除出工作集至磁盘等地

关于物理磁盘（PhysicalDisk）、网络接口（Network Interface，即网卡）和整个系统（System）的若干重要计数器以及含义如表 6-4 所示。

表 6-4　Windows系统下物理磁盘、网络接口和整个系统各计数器的含义

性能对象名称	计数器名称	计数器描述
PhysicalDisk 物理磁盘	%Disk Time	指所选磁盘驱动器忙于为读或写入请求提供服务所用的时间百分比
	Avg. Disk Queue Length	指读取和写入请求（为所选磁盘在实例间隔中列队的）的平均数
	Avg. Disk Read Queue Length	指读取请求队列的平均数
	Avg. Disk Write Queue Length	指写入请求队列的平均数
	Disk Reads/sec	物理磁盘上每秒钟磁盘读取操作的次数
	Disk Writes/sec	物理磁盘上每秒钟磁盘写入操作的次数
	Avg. Disk sec/Read	指以秒计算的在磁盘上读取数据所需的平均时间
	Avg. Disk sec/Transfer	指以秒计算的在磁盘上传送数据所需的平均时间
Network Interface 网络接口	Bytes Total/sec	为发送和接收字节的速率，包括帧字符在内
System 系统	File Data Operations/Sec	指在计算机的所有逻辑磁盘上读取和写入操作的综合速度。本计数器的数值不包括文件系统操作，后者对于 Web 应用并不重要
	Processor Queue Length	处理器队列的线程数量。对于系统中所有的处理器都使用单一队列（线程在该队列中等待处理器进行循环）。此长度不包括当前正在执行的线程

在这里，我们只是列出了性能计数器的名称和含义，对于在具体的场景中该如何分析这些计数器的数值并未涉及。这是因为在后面的章节中，还要介绍 Mac OS X 和 Linux 的性能计数器，虽然各个系统的名称可能会有所不同，但是含义都类似，因此，在介绍完所有 3 个系统的计数器后，再来探讨分析的方法会具有比较好的通用性。

6.3　Mac OS X 系统的性能计数器

目前，业内主流 Mac OS X 的版本是 10.4.X 或者 10.5.X（X 代表阿拉伯数字，以区分小版本），本书以截至 2008 年 10 月份最新的 10.5.5 版本为例，讲述性能计数器的获得，对于其他版本基本与之类似。

6.3.1　Mac OS X 系统下性能计数器的直观获得

与 Windows 系统下的任务管理器类似，Mac OS X 也有一个友好的界面提供给用户，用于获取系统当前基本的计数器信息。它就是 Activity Monitor 程序。在 Mac OS X 桌面上单击“硬盘”（Macintosh HD，或其他类似名称）图标，在弹出界面的左边导航菜单中选择“应用程序”（Applications）项目，此时，在右边会出现该项目下的程序与文件夹列表，如图 6-15 所示。



图 6-15　选择 Mac OS X 硬盘中 Applications 下的 Utilities 文件夹

单击其中的“工具”（Utilities）文件夹图标，即可进入系统提供的若干实用工具列表，如图 6-16 所示。双击其中的“活动监视器”（Activity Monitor）图标运行。

运行后，将弹出类似 Windows 下任务管理器进程列表的界面，如图 6-17 所示。

任意选择进程列表中的一条记录，单击界面左上方的 Inspect（探查）按钮，就可以查看关于所选择进程的进一步信息，如图 6-18 所示。

以上是通过图形界面获取 Mac OS X 系统性能计数器的方法。实际上，还有通过 Shell 命令来获取它们的方法，由于 Mac OS X 系统本身是一个变种的 Unix，我们将在 6.4.1 节 Linux（Unix）系统下性能计数器获得方法中介绍，它们所使用的命令基本是一致的。

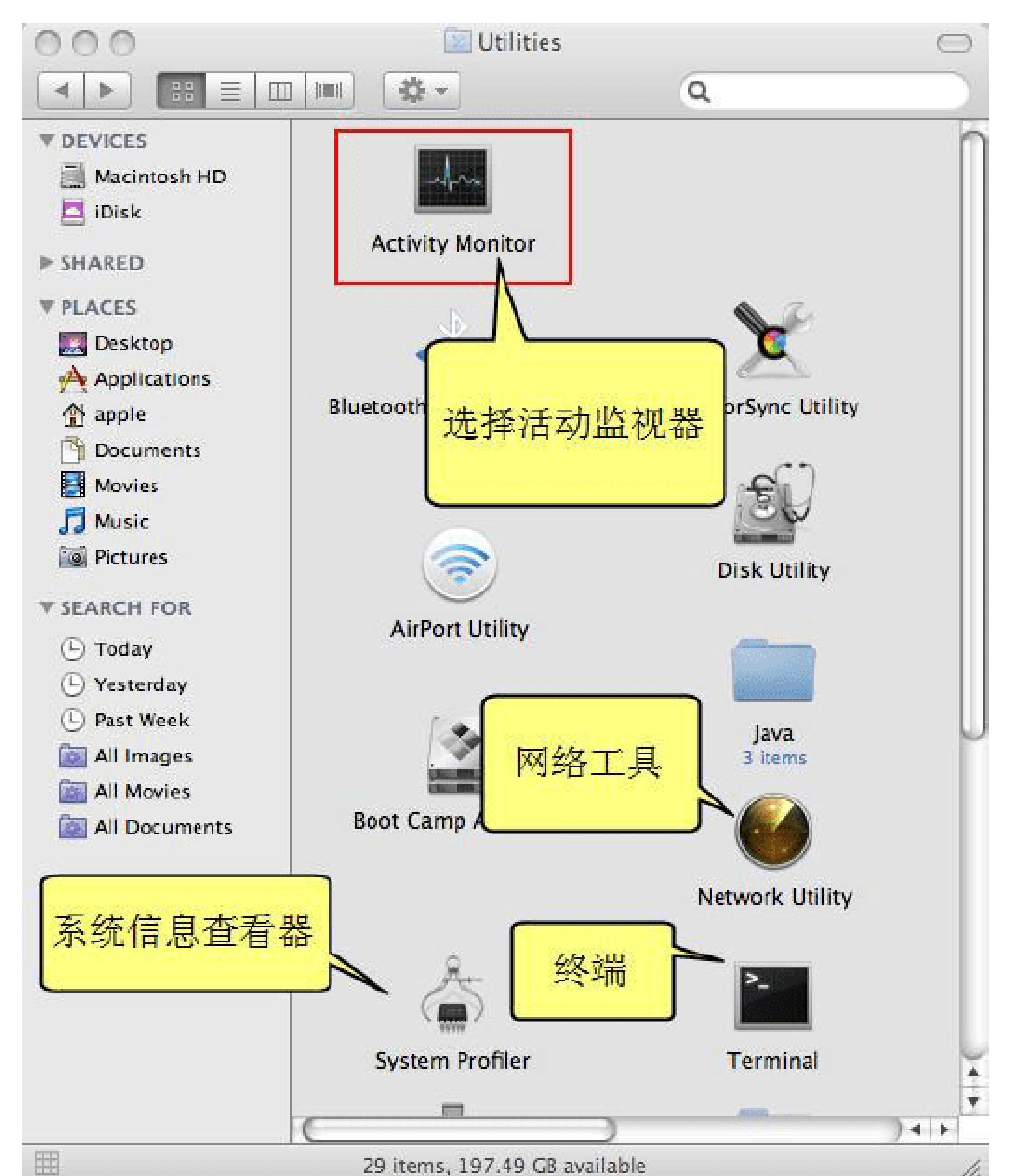


图 6-16　Mac OS X 的 Utilities 文件夹下包含诸多实用工具

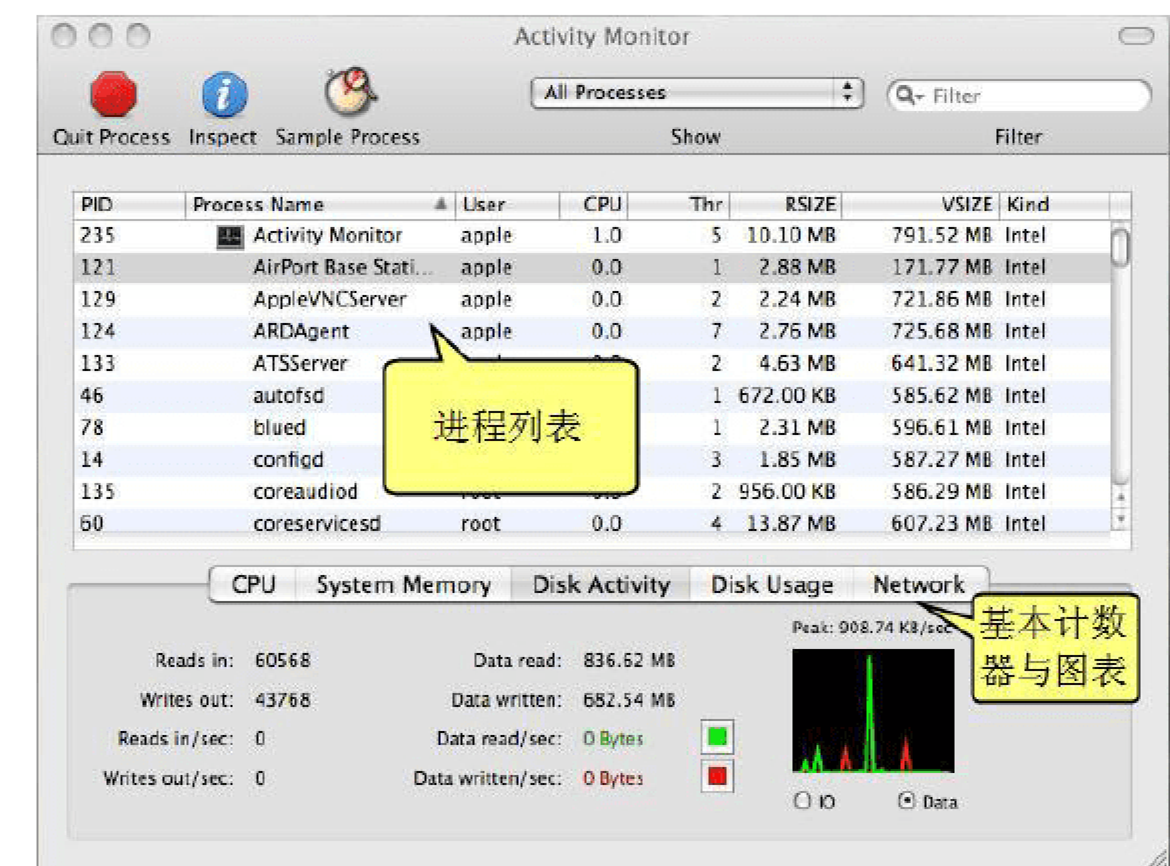


图 6-17　Mac OS X 系统下的活动监视器



图 6-18　获取某个进程的计数器信息

6.3.2　Mac OS X 性能分析专业工具

除了前文所讲述的系统自带 Activity Monitor 程序，还有其他的一些专业性能分析工具，它们相对于活动监视器的优点在于可以使得性能计数器采集数据自动化，便于自动化测试。比较常用的有两类：

- 苹果网站开发社区中开发人员提供的一些脚本，它们对于在 Mac OS X 环境下开展工作的测试工程师是很有用的。比如 top-guide.pl 这个 Perl 脚本可以将系统磁盘、网络 I/O 等信息输出，或者形成文件便于分析。苹果网站开发社区的网址为：http://www.apple.com/support。
- 专业性能分析与开发工具，比如 CHUD（在英文里是一种变种的地下怪兽），它也可以在上述的网址中找到。这个软件包内包含一系列的小工具，非常适合 Mac

OS X 下软件开发工程师和性能测试工程师使用，多用于代码级的白盒测试。如图 6-19 是 CHUD 软件包中用于性能分析的 Shark（英文中是鲨鱼的意思）程序运行结束后的报表界面。

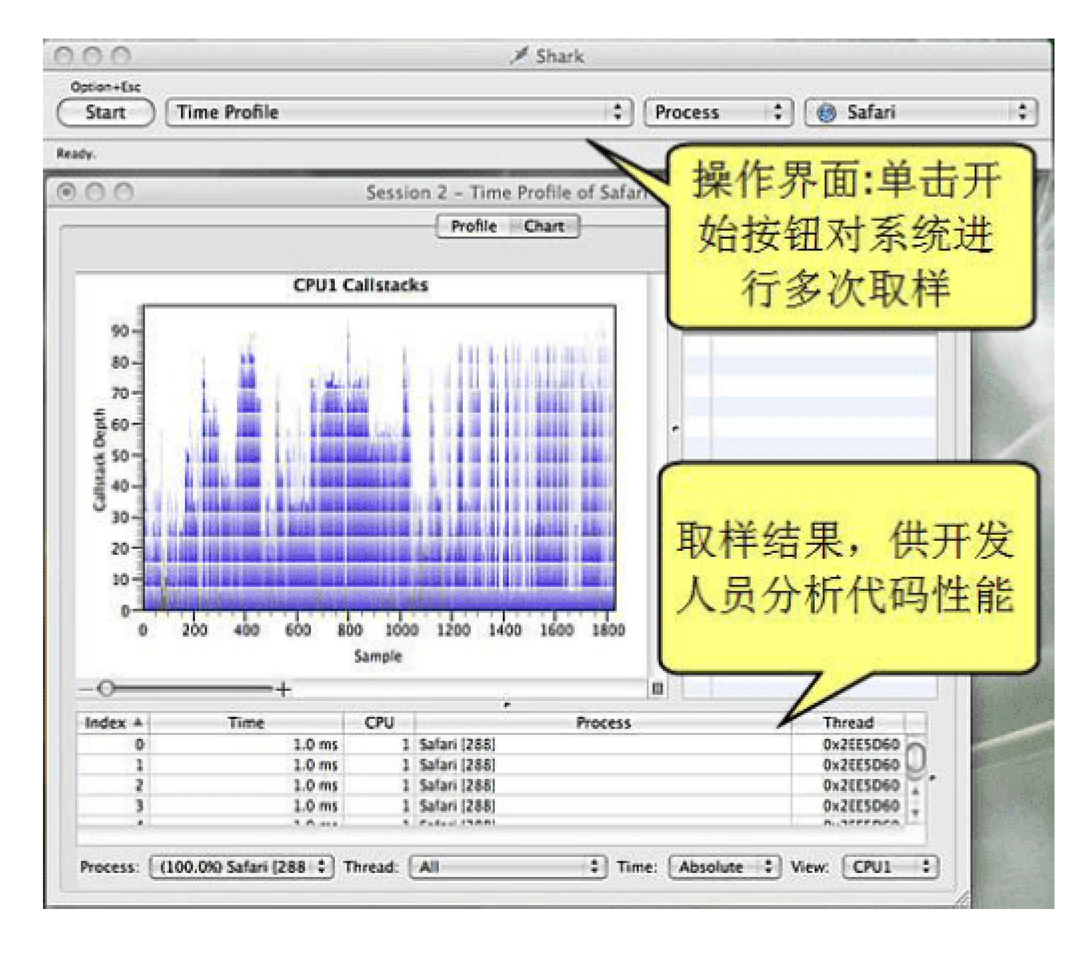


图 6-19　CHUD 软件包下 Shark 工具某次的运行结果

6.4　Linux（Unix）系统的性能计数器

由于 Linux 相对于 Unix 更容易获得，而且两者在获取系统计数器方面都很相近，这里以 Linux 系统为例。读者如果在工作中遇到了 Unix 系统，可以查询相关的操作手册。

6.4.1　Linux 系统下性能计数器的直观获得

在笔者 Linux 的 Gnome 桌面中，提供了类似 Windows 任务管理器的一个系统监视程序，可以用于对整个系统做个粗略的、图形化的感官认识。它的打开方式是依次选择“系统”|“系统工具”|“系统监视”命令，弹出界面如图 6-20 所示。

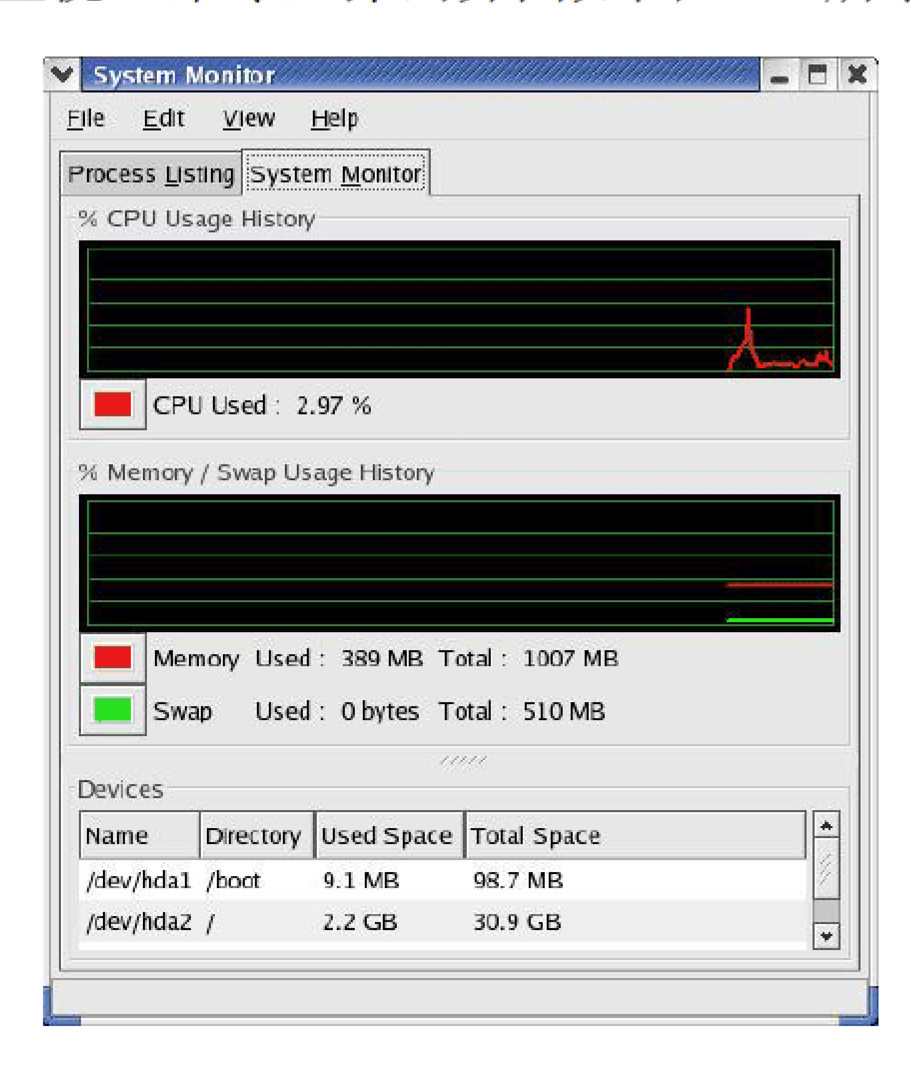


图 6-20　Gnome 桌面下的系统监视

【字符界面工具更为常用】

不过，对于服务器上的 Linux 来说，一般都使用字符界面，因此我们需要通过若干个 shell 命令来获取计数器信息，它们分别是 vmstat、top、sar、iostat 等。

6.4.2　vmstat 命令详解

vmstat 命令的全称是 virtual memory statistics，即反映当前系统虚拟内存（当然并不局限于此）的使用情况。

在 Linux 的“终端”（Terminal）程序中输入 vmstat 然后按下 Enter 键，默认的输出举例如图 6-21 所示。为了清楚起见，笔者把输出的各类别（进程、内存、交换分区、块设备、系统和 CPU）用方框标了出来，实际显示是没有的，因此有的初学者可能会看不明白具体各项数据的所属。

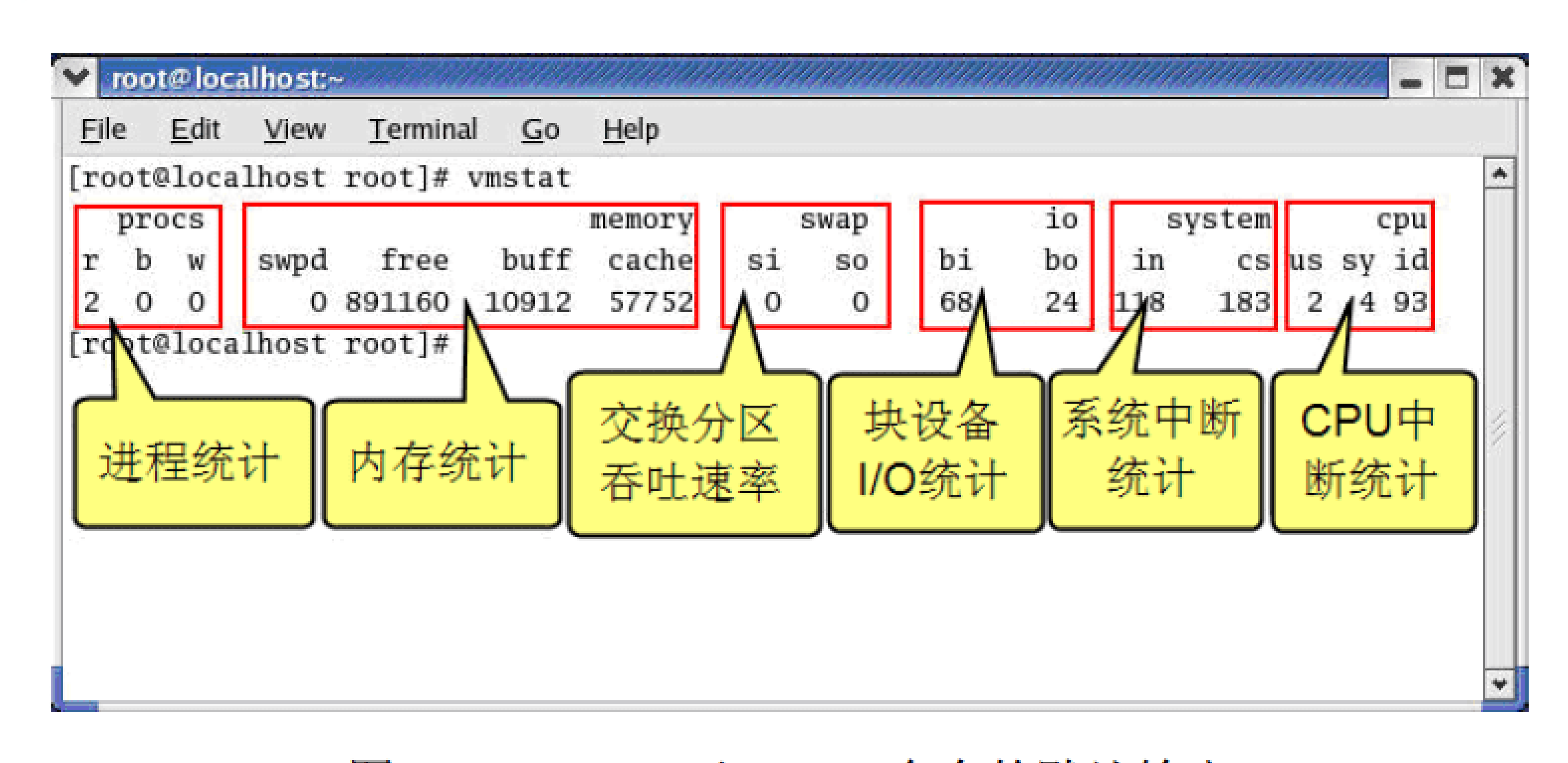


图 6-21　Linux 下 vmstat 命令的默认输出

在各方框的第 1 行都是类别，类似 Windows 下的性能对象。第 1 行是具体的计数器，第 3 行是它们各自的值。如果在 vmstat 后面跟上时间开关，还可以获得一定时间内的计数器数据。

下面是图 6-21 中各计数器的含义，在 6.4.4 节中还要专门以列表的形式介绍主要的 Linux 计数器。

1．进程（Procs）

- r：等待运行的进程数。
- b：处在非中断睡眠状态的进程数。
- w：被交换出去的可运行的进程数。

2．内存（Memory）

- swpd：虚拟内存使用情况，单位：KB。
- free：空闲的内存，单位 KB。
- buff：被用来做为缓存的内存数，单位：KB。
- cache：被用来作为数据缓冲区的内存数，单位：kB。

3．交换分区（Swap）

- si：从磁盘交换到内存的交换页数量，单位：KB/秒。

- ❑ so：从内存交换到磁盘的交换页数量，单位：KB/秒。

4. 输入输出（I/O）

- ❑ bi：发送到块设备的块数，单位：块/秒。
- ❑ bo：从块设备接收到的块数，单位：块/秒。

5. 系统（System）

- ❑ in：每秒的中断数，包括时钟中断。
- ❑ cs：每秒的环境（上下文）切换次数。

6. CPU，按 CPU 总使用时间的百分比来显示

- ❑ us：CPU 用户使用时间。
- ❑ sy：CPU 系统使用时间。
- ❑ id：CPU 闲置时间。

6.4.3　top 命令以及其他工具包

Top 命令也可以用来查看内存、进程等信息。在系统的“终端”（Terminal）程序中输入 top 并按下 Enter 键，默认的输出举例如图 6-22 所示。Top 命令能够显示当前系统中按照%CPU 等计数器数值排序的进程列表，并且实时更新，具体如图 6-22 所示中的下半部分。

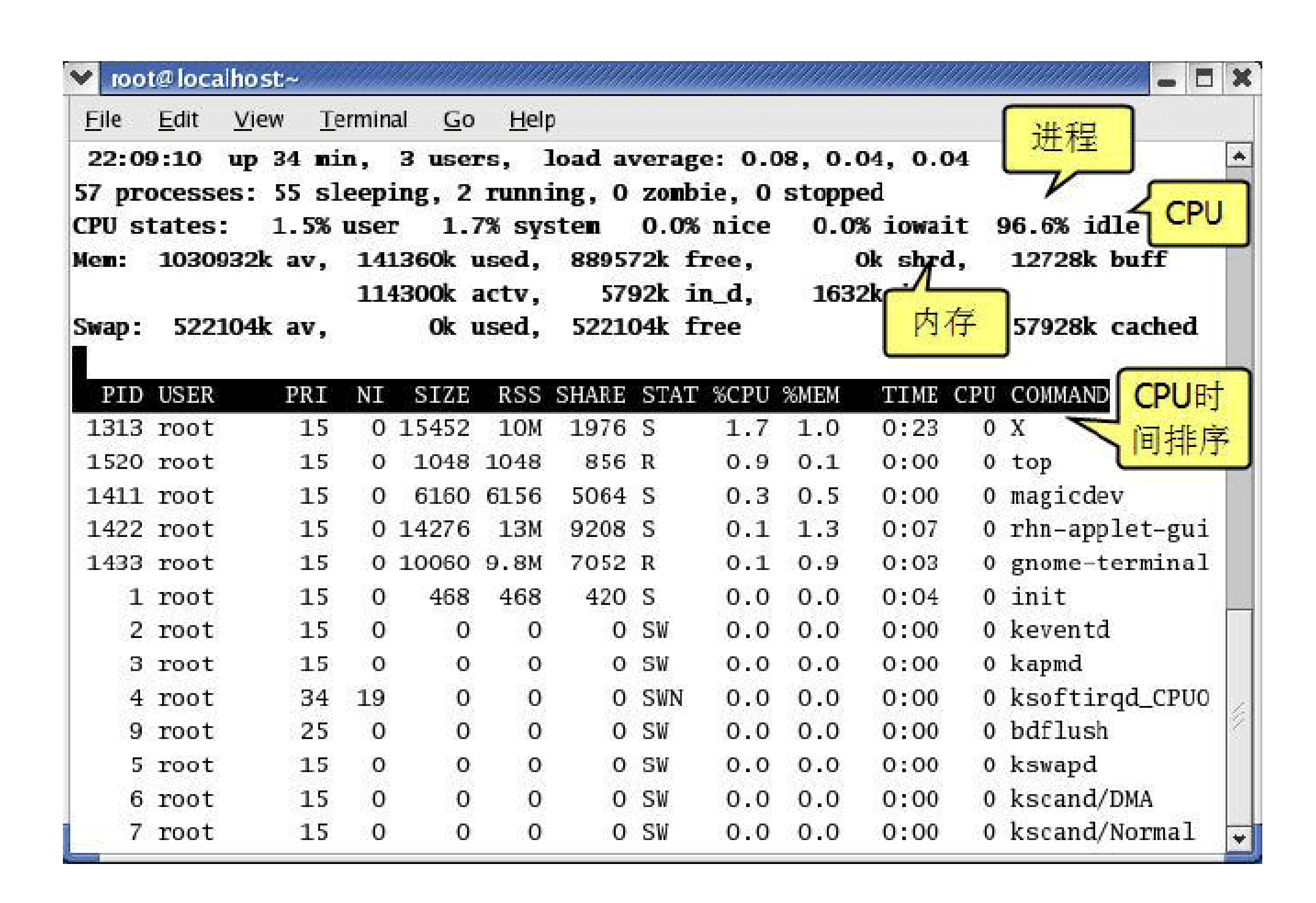


图 6-22　Linux 系统下 top 命令的默认输出

【Sysstat 工具包】

除了以上两个命令之外，有的 Linux 版本还提供了 Sar、iostat 等有用的命令，它们还支持记录成日志文件、转化为数据库记录等多种功能。如果读者在自己的 Linux 中输入 Sar，提示命令找不到，则可以到 http://linux.softpedia.com/get/Utilities/Sysstat-4621.shtml 这个网址下载并安装 Sysstat 工具包。截至 2008 年 9 月，最新版本是 8.1.6。Sysstat 工具包中的各

命令具体使用方法请参考命令帮助。

由于 Unix 和 Linux 发行版本多种多样，每个版本都会有自己特别的一些命令以提供使用上的方便，比如前面提到的 Mac OS X 10.5 就增加了查看线程锁的命令 plockstat 等。我们在进行性能分析前要多阅读当前版本的管理员手册。

6.4.4　Linux（Unix）系统下性能计数器的含义

我们依然用表格来列出 Linux（Unix）系统下主要性能计数器的含义，这也包括前文所提到的 Mac OS X 系统，如表 6-5 所示。注意，根据不同的发行版本，具体计数器的名称可能有所不同。

表 6-5　Linux（Unix）系统主要性能计数器的含义

类　别	计数器名称	计数器描述
Memory 内存	Free （KB）	可用物理内存数
	Swap （KB）	已使用的虚拟内存数量
	（page） si	每秒从磁盘交换到内存的数量
	（page） so	每秒从内存交换出的内存数量
	Cache （KB）	文件系统缓存
Process 进程	%CPU Usage	进程消耗的处理器时间
	Page Fault count	该进程产生的页面失败次数
	Resident size （KB）	进程保留的使用内存量。该数值等于进程的代码使用内存＋进程的数据使用内存
Processor 处理器	%Idle Time	Idle Time%描述的是 CPU 总的空闲时间。如果该值持续低于 10%，表明瓶颈是 CPU。可以考虑增加一个处理器或换一个更快的处理器
	%User Time	非内核操作耗费的 CPU 时间
	%Kernel Time	CPU 内核时间是在核心模式下处理线程执行代码所花时间的百分比
	%IO wait Time	CPU 消耗在等待 I/O 处理上的时间，此值需要结合 I/O 的计数器考虑
Physical Disk 物理磁盘	Average number of transactions actively	指读取和写入请求（为所选磁盘在实例间隔中列队的）的平均数
	Percent of time the disk is busy	指所选磁盘驱动器忙于为读或写入请求提供服务所用的时间的百分比
	being serviced Average number of transactions waiting for service	指读取（写入）请求（列队）的平均数
	Reads（Writes） per sec	物理磁盘上每秒磁盘读、写的次数。两者相加，应小于磁盘设备最大容量。在 iostat 的结果中，该值显示为 r/s 和 w/s
	Average service time active transactions, in milliseconds	指以毫秒计算的在磁盘上读取和写入数据的所需平均时间。在 iostat 的结果中，该值显示为 asvc t
	The number of disk operations per second	显示每个磁盘每秒的被操作次数
System 系统	%User Time	系统上所有处理器执行非内核操作的平均时间的百分比
	CPU context switches	处理器上下文切换次数

从表 6-5 中也可以看出，不同的操作系统计数器名称有所区别，但实际含义相差不大。我们完全可以从一个操作系统出发，慢慢在工作中熟悉更多操作系统计数器的含义。这也是学习操作系统知识的机会。

6.5　内存性能分析

在前几节中，我们介绍了很多操作系统的计数器，那么，它们的具体数值与常见的性能问题有什么联系呢？也就是说，如何从某个计数器的表现，来判断系统当前的内存性能状态呢？本节将介绍这方面的内容。

判断内存性能表现主要是为了解决如下两个问题：

- ❑ 判断当前 Web 应用是否存在内存泄露的问题，如果有，问题的程度有多大。
- ❑ 如果 Web 应用的代码无法进一步改进，当前 Web 应用所在的服务器是否存在内存上的瓶颈，即是否需要增加内存数量来提高性能。

6.5.1　内存泄露及判断

首先要简单介绍一下内存泄露以及对其的判断。对于从事黑盒测试的工程师来说，这部分内容有一点点深度，而且在实际工作中用处并不很大（内存泄露往往是开发工程师进行单元测试、资深性能测试工程师进行深入的性能测试时所关注的问题），但需要有这方面的概念和意识。

1．内存泄露

还记得前面章节中提到的内存泄露吗？我们用饭馆中发生的事情进行了类比。服务员在客人吃完饭后没有收拾桌子，因此，后面的客人无法使用该餐桌，导致饭馆实际可容纳的客人减少。

内存泄露与此类似。程序都需要装载在内存中才能运行，退出时候，将内存返还给系统。这种返还有两种方式：

- ❑ 自行返还。由程序员在编写代码的时候返还，比如 C++的代码等。
- ❑ 自动返还。由支持程序运行的“服务”平台，比如 Java、.NET 等定期自动返还。

但是，在有的时候，无论是哪种返还方式，程序都可能未把自己使用的内存还给系统，因此造成之后的程序无法再使用这一部分内存，好像从下水道漏掉消失了一样。一次性的内存泄露危害不大，但是对于 Web 应用所在的服务器场景来说，网站程序都是持续不间断地运行的。如果代码存在内存泄露的问题，是会堆积起来，越来越严重的。显而易见，这种堆积型的内存泄露导致了系统可使用的内存资源持续变少，系统性能会变得越来越差，对于用户所感觉到的网站性能表现来说，就是响应时间的变慢甚至部分网站功能的失效。

2．性能计数器与内存泄露

判断内存泄露需要性能计数器的帮助。为了简单起见，我们以 Windows 为例，在系统的 Perfmon，即性能程序中需要关注 Process（进程）性能对象的如下几个计数器：

- ❑ Handle Count（句柄数量）；
- ❑ Thread Count（线程数量）；
- ❑ Virtual Bytes（虚拟内存字节数）；
- ❑ Working Set（工作集，即和当前进程有关的那一部分物理内存）；
- ❑ Private Bytes（进程分配的私有数据字节数量）。

一般情况下，虚拟内存字节数应该远大于工作集字节数，如果两者变化规律相反，比如说工作集增长较快，虚拟内存增长较少，则可能说明出现了内存泄露的情况。对于所有这些性能计数器，如果在测试期间内数值持续增长，而且测试停止后维持在高水平，则也说明存在内存泄露。

3．内存泄露相关专业软件

通过计数器一段时间的变化规律，可以粗略地判断是否有较严重的内存泄露发生。为了尽量避免这样的内存泄露给系统性能造成的危害，从源头，即开发工程师编写程序时进行控制是比较好的办法。有不少相关的专业软件，比如 Quest 公司的 JProbe 套件、IBM 的 Rational Purify、微软的 leakdiag、BoundsChecker、CLRProfiler（将在第 17 章进行介绍）等可以使用，开发工程师、性能测试工程师可以在代码不那么庞大的时候就进行单元测试、组件测试，不要让问题积累起来。

内存泄露是一个比较复杂的问题，需要很多方面的知识，有兴趣的读者可以阅读专业书籍进行学习。这里只是介绍一些简单的理念，在工作中能够有这样的意识发现问题。

6.5.2　内存瓶颈简介

所谓内存瓶颈，是指由于可用内存的缺乏导致系统性能下降的现象。瓶颈英文为 Bottleneck，看到它总有一种被人掐住脖子的感觉，因此不是很舒服。

1．内存瓶颈相关性能对象

内存瓶颈相对内存泄露要好理解和容易发现得多。分析系统是否存在内存瓶颈，主要考虑内存的页面操作和磁盘的 I/O 操作。若我们以 Windows 为例，则相应需要考虑如下的性能对象：

- ❑ Memory 性能对象，主要用于分析整个系统的内存瓶颈问题。
- ❑ Process 性能对象，主要用于分析某个应用的内存瓶颈问题。
- ❑ Physical Disk 性能对象，与内存瓶颈被动相关，将在后面的 6.7 节中提到。

下面列举几个用于判断内存瓶颈的重要性能计数器。

2．几个重要性能计数器

前述这两个性能对象中有几个计数器需要特别关注，它们分别是：

- ❑ Available MBytes（或者 Kbytes 等几个单位不同、含义一样的 Memory 性能计数器）。该计数器的值显示了当前系统可用的内存数值。如果经常性的可用内存小于总物理内存的一半，说明系统负担有点重，可以考虑增加内存。
- ❑ Pages/sec、Pages Input/sec、Pages Read/sec 与 Page Faults/sec。这几个计数器在

Memory 性能对象和 Process 性能对象中都有，分别代表整个内存和单个进程（程序）的页面操作信息，可以用于分析整个系统或者关注的程序使用内存的状况。

判断内存瓶颈，必须要掌握页面的相关知识。

6.5.3　页面和虚拟内存

所谓页面（Page），就是 CPU 对物理内存进行管理的一个单位，一般为 4k 大小。页面内一般放置的是程序代码和数据。

前文中提到了工作集的名词，它是和当前进程相关的那一部分内存的总称。页面和工作集是两个概念，一个工作集包括多个页面。但是，光有工作集是不一定够用的，因为对于我们的计算机来说，内存总是有限的，而需要使用内存的程序非常多，大家都把自己需要的代码和数据全部放在内存中是吃不消的。因此，操作系统会在磁盘上也为程序开辟一块存储的地盘，当做内存使用，这也就是虚拟内存。虚拟内存在磁盘上是以文件形式存在的，在 Windows 中文件名称叫做 PageFile.sys，如图 6-23 所示。

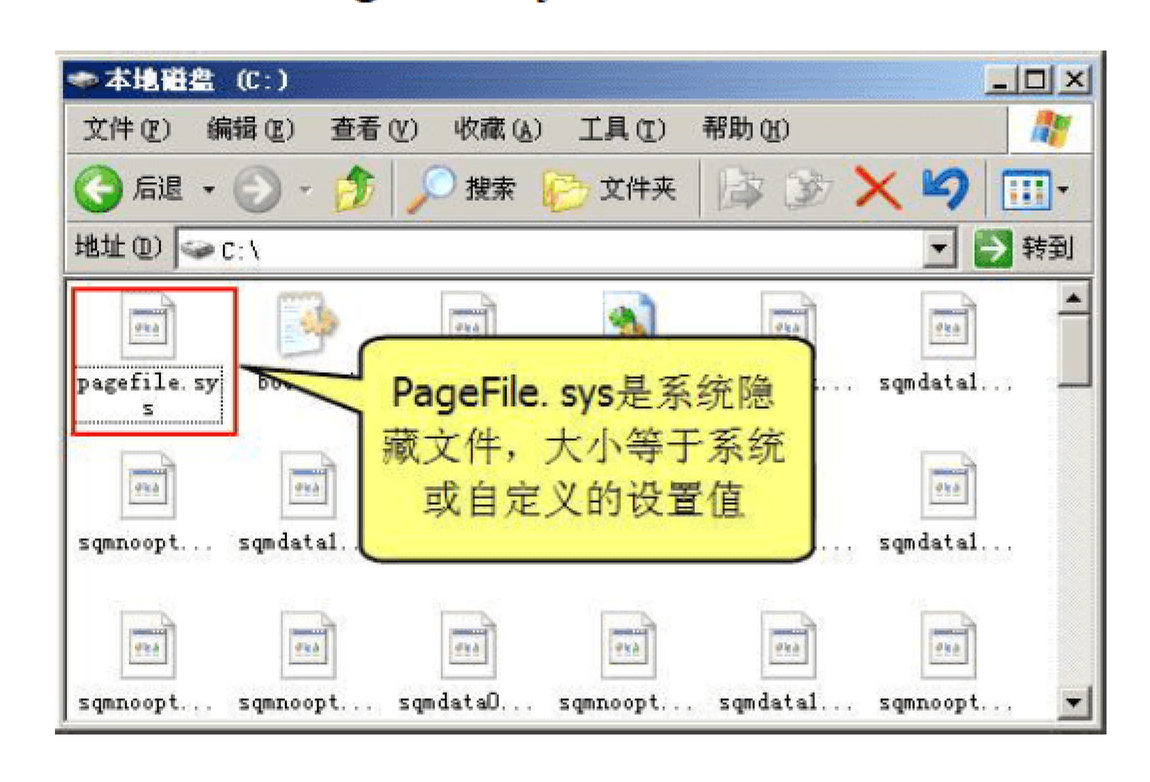


图 6-23　某电脑 C 盘根目录下的 PageFile.sys 文件

虚拟内存的具体位置和大小可以设置：右击桌面上的“我的电脑”图标，按照图 6-24 所示的操作顺序进行更改。

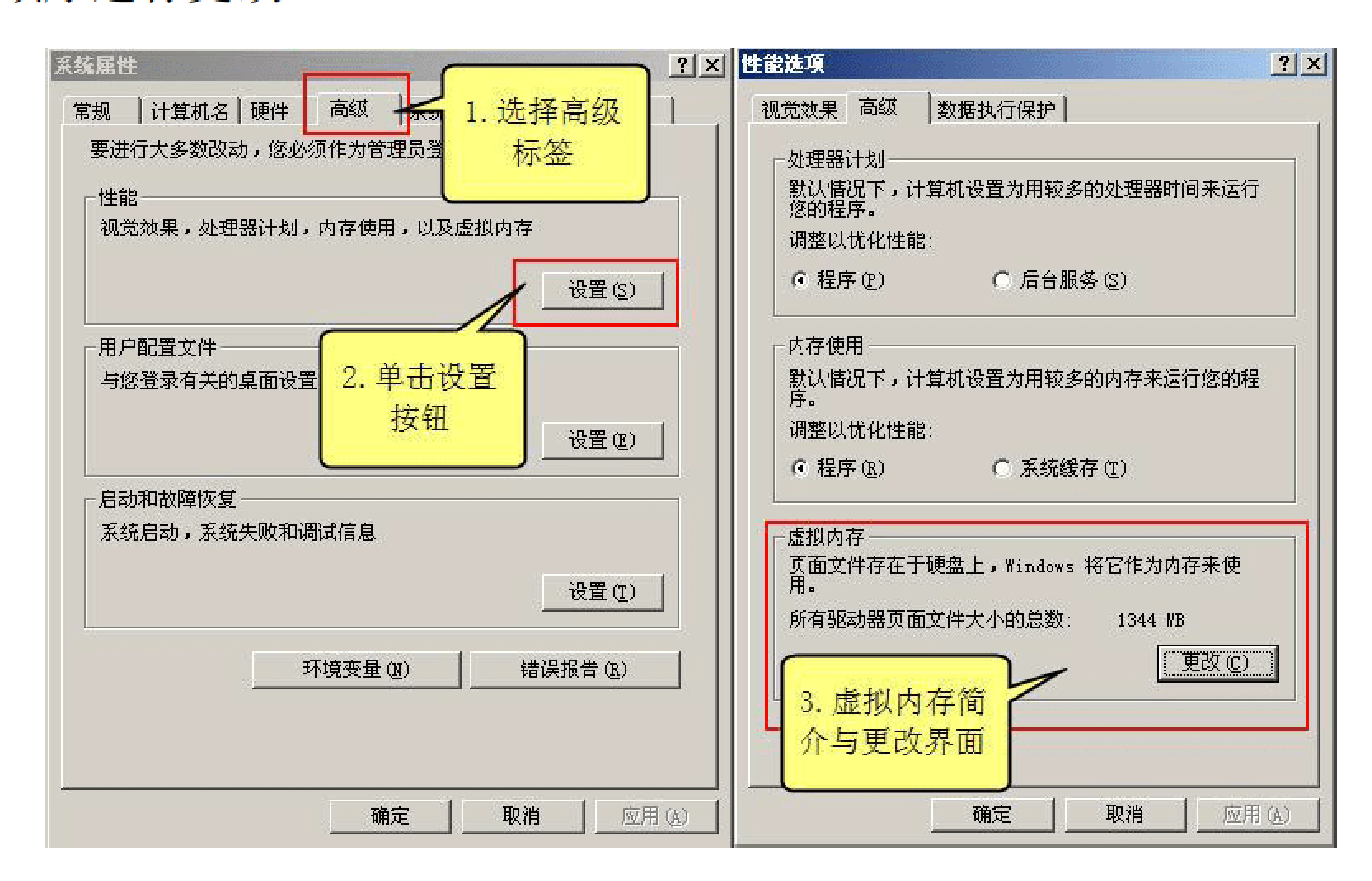


图 6-24　打开设置当前系统虚拟内存的步骤

【页面的传递】

程序在运行中所需要的数据是可以变化的。举一个形象但不一定恰当的例子，在某一时刻，界面需要某个按钮；而另一时刻，这个按钮就不再需要，那么还把这个按钮保留在工作集中就有点浪费，不如把它转移到虚拟内存中。这样做的原因是内存访问起来速度比磁盘快很多，为了程序使用方便，操作系统经常要相互移动磁盘与内存中的数据，这个过程也叫做页面的传递（Paging），与人们在换季时将卧室衣柜里的过季衣服转移到箱子的过程有些类似。

总之，这些数据的转移过程是非常复杂的，作为初学者，读者没有必要深究。若用简单、通俗的语言来描述，就是把程序近期用得着的数据放在内存里，用不着的数据放在虚拟内存，也就是磁盘的某一块区域中。这样能提高程序访问数据的速度，从而提高程序运行速度。

下面将简要介绍判断内存瓶颈时最有用的线索，页面错误的相关知识。

6.5.4　软、硬页面错误

综上所述，对于一个运行中的程序来说，它所需要的代码和数据可能在几个地方找到，如图 6-25 所示。

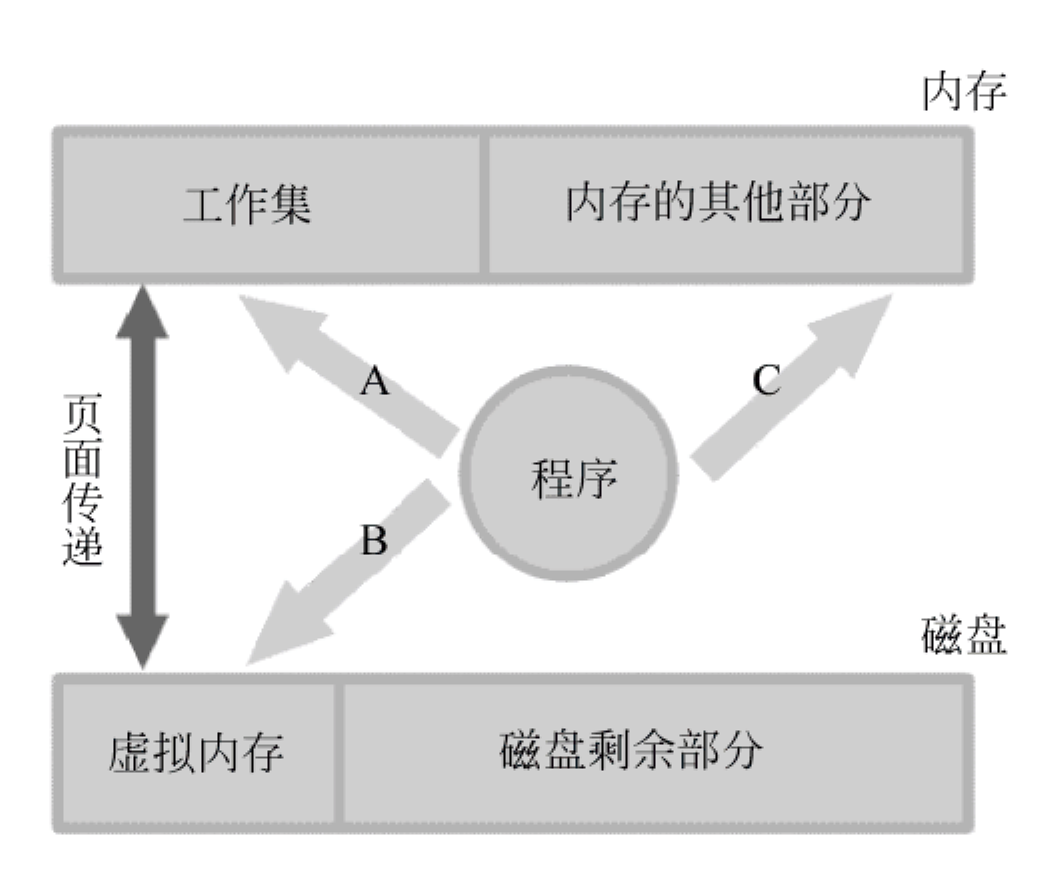


图 6-25　运行时程序代码和数据可能存放的位置

在图 6-25 中，程序得到所需要的代码和数据共有 3 条线路。

- 线路 A，程序访问工作集。由于工作集是内存中与当前程序相关的一部分，速度最快，不会对系统性能造成什么坏的影响。
- 线路 B，程序访问虚拟内存。虚拟内存实际是硬盘的一部分，硬盘速度要远小于内存速度，因此这部分速度最慢，处理不好，可能会对系统性能造成严重影响。如果程序所需要的数据通过线路 A 找不到，必须在线路 B 中才能找到，则我们把这种情况叫做硬页面错误（Hard Page Fault）。
- 线路 C，程序访问内存的其他部分。有些数据是存放在这里的，它可能与当前程序相关，但不被其独享。如果程序所需要的数据通过线路 A 找不到，必须在线路 C 中才能找到，则我们把这种情况叫做软页面错误（Soft Page Fault）。

在这 3 条线路中，可能对系统性能造成最大影响的是线路 B，即硬页面错误。它也是判断内存出现瓶颈最有用的指标。

6.5.5　发现内存瓶颈

有了前文所讲述的背景知识，理解发现内存瓶颈的方法就比较简单了。主要步骤如下。

（1）打开 Windows 性能程序（Perfmon.exe），按照前文的方法增加 Memory 性能对象下面的 4 个计数器。

- Pages Input/sec，程序在硬页面错误发生时从虚拟内存（磁盘上的页面文件）中每秒获得所需数据页面的数目。
- Pages/sec，是每秒 Pages Input 和 Page Output（程序向虚拟内存写数据的页面数目）的和。
- Page Faults/sec，程序每秒钟在工作集中找不到所需要数据的次数，相当于图 6-19 中的线路 A 失败，因此，程序将从线路 B 和线路 C 出发去寻找，该数值等于硬页面错误次数和软页面错误次数相加之和。
- Page Reads/sec，程序通过线路 B 读取虚拟内存（即硬盘上的页面文件）的次数（注意，这里是次数，而不是页面数目）。平均数值长时间超过 5，表明内存存在瓶颈。

（2）如果 Page Reads/sec 数值经常性地超过 5，或者 Pages Input/sec 比较大，比如平均值到达 10 或者更多，或者 Page Input 和 Page Faults 之间的比率超过 50%，都说明内存给系统性能带来了问题。

（3）如果 Page Reads/sec 数值上比 Page Input/sec 要大，说明系统 1 秒钟读取了多个页面，也说明内存会给系统性能带来问题。

6.5.6　发现程序使用内存的问题

前面所讲述的方法都是观察整个系统内存的瓶颈，那么，如何发现具体某一个程序对于内存的影响呢？测试工程师可以利用 Process 性能对象下的类似计数器来查看。具体方法如下。

打开 Windows 性能程序（Perfmon.exe），按照前文的方法增加 Memory 性能对象下面的 Page Faults/sec 计数器，然后选择 Process 性能对象，增加不同 Process 的 Pages Faults/sec 计数器，如图 6-26 所示。

笔者在这里分别增加了 IE 浏览器、IIS 服务、Photoshop 等程序的页面错误计数器后，性能图表如图 6-27 所示。

从图 6-27 中可以得出如下几点结论。

- 如果类似这样的直方图持续很长时间，则说明数值最高的那个程序（注意，在图 6-26 中是第 2 个柱体）比其他程序更容易造成页面错误，可以查看该程序代码以发现内存管理方面的问题。
- 如果图中代表程序页面错误最大数值的那个柱体与内存中所有页面错误的柱体相差较大（类似图 6-26 这样的情况），则说明当前并没有将对内存影响最大的程序包括进来。也就是说，并没有发现造成系统页面错误较多的主要因素。

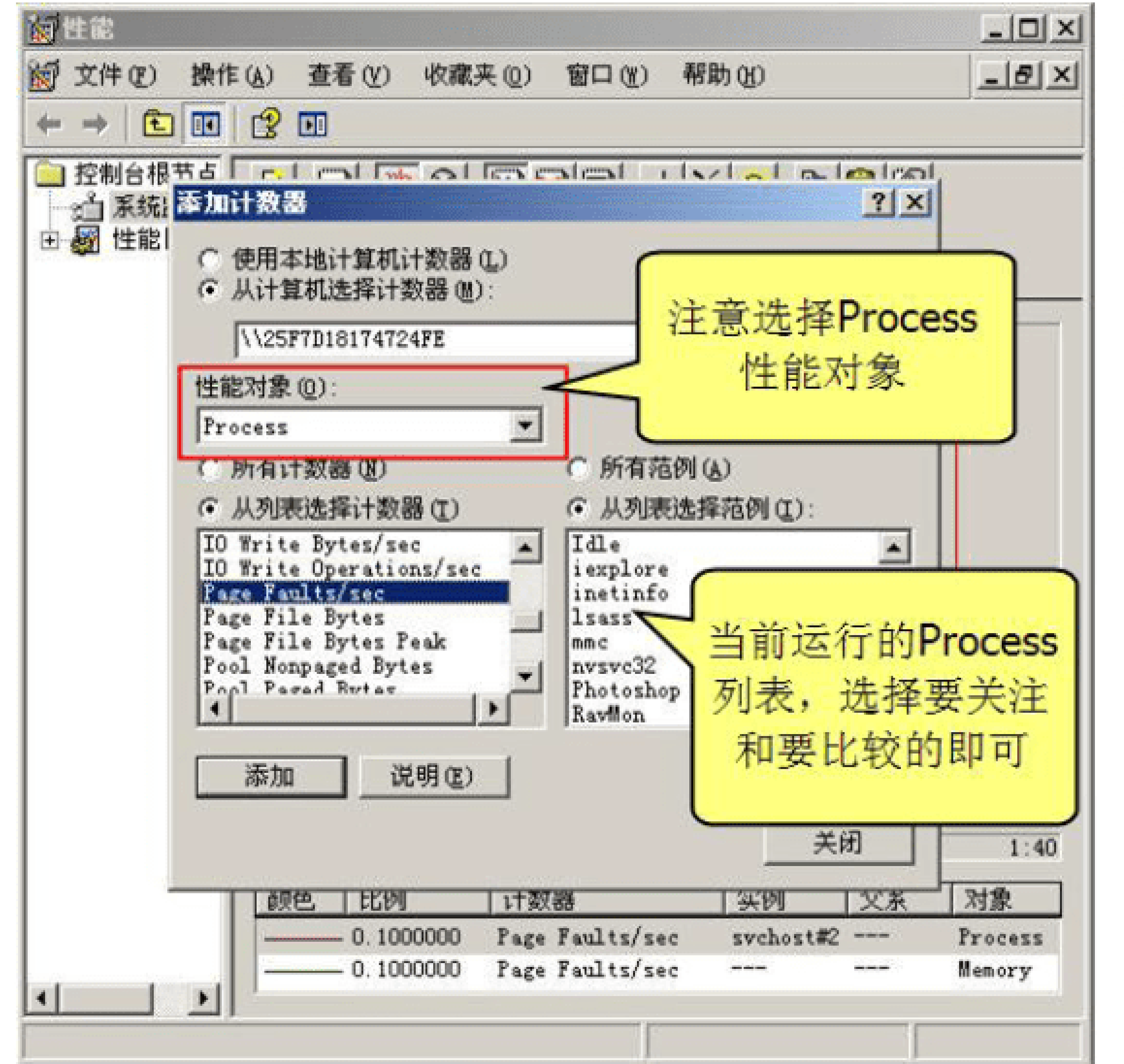


图 6-26　增加某程序页面错误计数器的方法

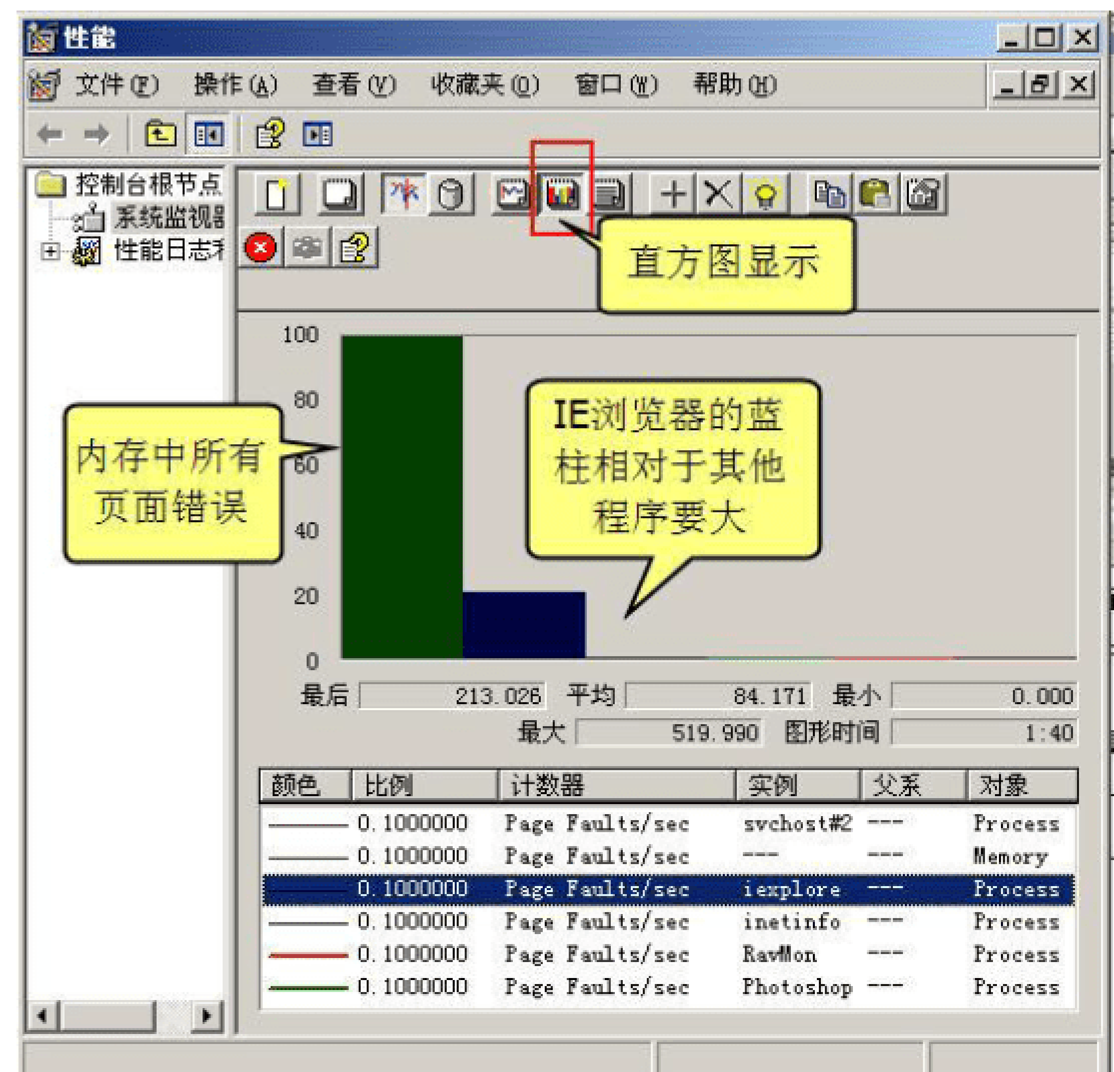


图 6-27　多个程序造成内存页面错误的比较直方图

通过观察一段时间后，选定程序页面错误数量和内存总页面错误数量的比较，可以发现哪些程序在内存管理上有不足的地方。如果在代码中以目前所知不存在不足，则说明系统内存需要扩展。

上述这种方法只能确定具体哪个程序在内存方面的问题，如果需要深入探讨，需要一系列的专业工具，比如 Page Fault Monitor（Pfmon.exe）等，它们可以提供函数级别的页面错误监视。对于其他非 Windows 系统，也都提供了很多类似功能的计数器，比如，在 Unix/Linux 系统中，有：

- ❑ Page Ins/sec
- ❑ Page Outs/sec
- ❑ Page Faults/sec

等，它们和 Windows 下的性能计数器有名称上的微小差别，原理和使用方法都是类似的。

【自动化的挑战】

无论是利用 Windows 性能监视器程序，还是 Unix 下的命令获得计数器长时间的变化数值，在自动化方面都比较麻烦，因此需要相关软件工具，能够定时帮我们做获取数据等工作，最好还能支持多种操作系统平台，做到同时获取。在后面的章节中，我们将学习如何利用 Load Runner 来获取这些内存相关计数器的数值。

6.6　CPU 性能分析

相对于内存性能会收到各种程序的复杂影响，CPU 的性能分析则相对比较简单。我们依然以 Windows 系统为例。

与 CPU 相关的性能计数器主要处于以下几个性能对象中：

- ❑ Processor（处理器）性能对象；

- ❑ Process（进程）性能对象；
- ❑ System（系统）性能对象；
- ❑ Thread（线程）性能对象。

6.6.1　重要的 CPU 性能计数器

重要的性能计数器及判断出现瓶颈的标准有：

（1）Process 性能对象下的处理器时间百分比% Processor Time（_Total），一般平均不要超过 70%，最大不要超过 90%。

这个百分比表明了处理器真正在负荷下工作的时间，等于处理器 Processor 性能对象下的% Processor Time（_Total）（即图 6-28 中绿色柱子的部分）减去系统空闲进程所耗费的时间百分比，在实际使用中不要混淆，如图 6-28 所示。

（2）Processor 对象下的%User Time 与%Privileged Time。对于一个程序来说，其所花费的处理器时间应该有如下关系：

```
%Process Time = %User Time + %Privileged Time
```

所谓 User Time 和 Privileged Time，实际上就是指程序分别处于操作系统的 User 模式和 Kernel 模式的时间。操作系统的 I/O 和系统服务一般运行在 Kernel 模式，程序一般情况下在 User 模式下运行，但对系统的调用处于 Kernel 模式下，如图 6-29 所示为两种模式的示意图。

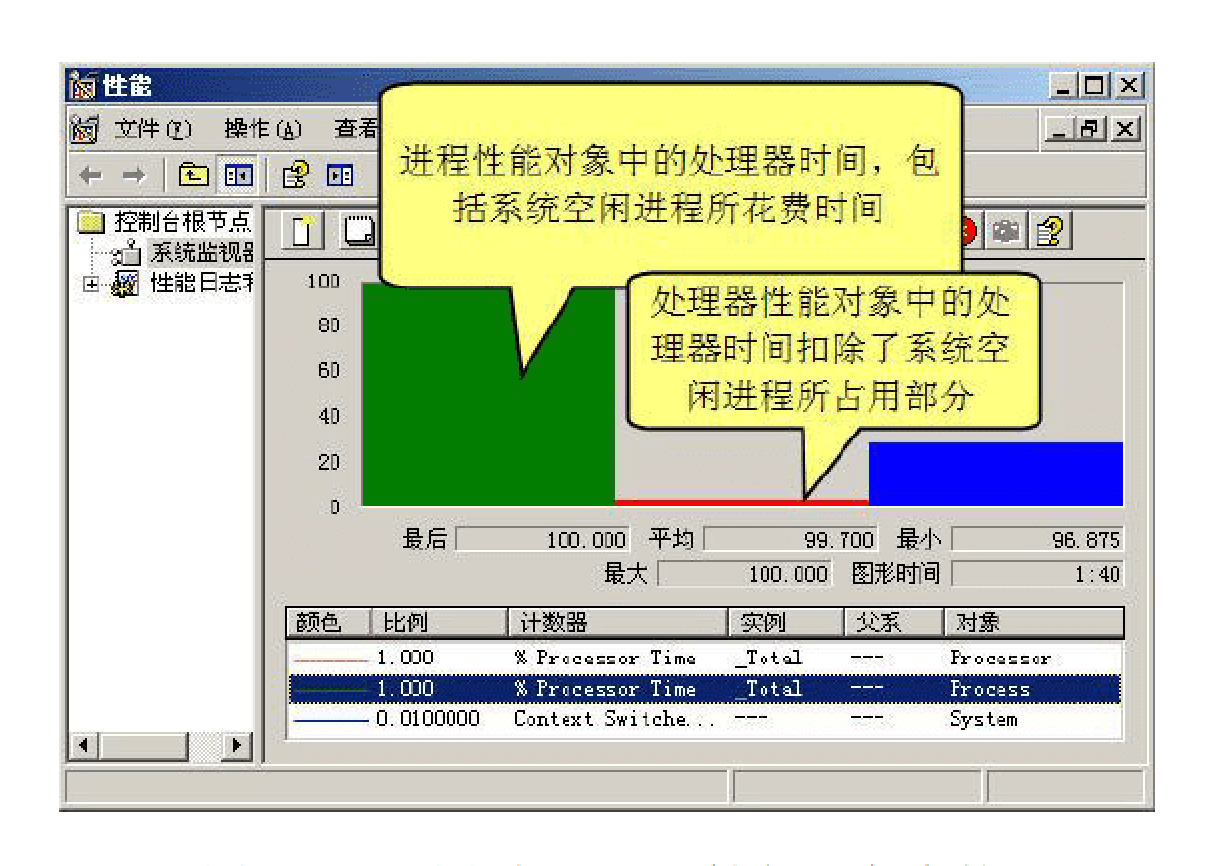


图 6-28　进程与处理器性能对象中的% Processor Time（_Total）区别

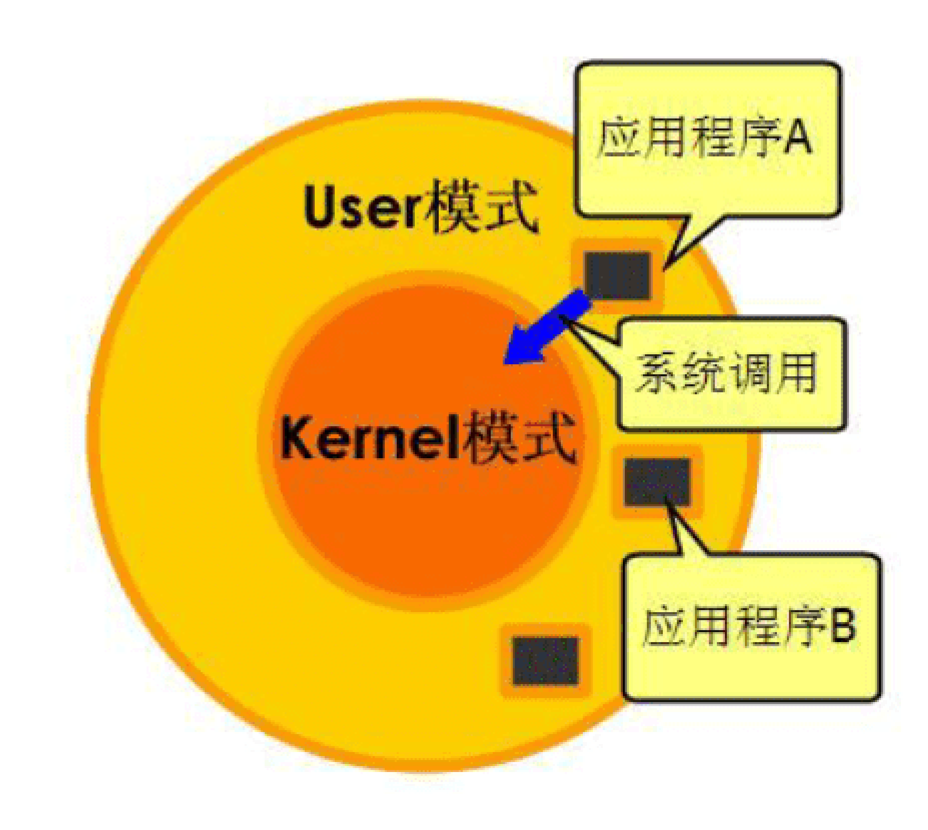


图 6-29　操作系统的用户模式与 Kernel 模式

大部分情况下，%Privileged Time 要远小于%User Time。因此，%User Time 在很大程度上代表了%Processor Time，同样它也不应过大，否则应该查看应用程序代码，对其算法、数据结构等进行优化。

（3）System 系统对象下的每秒上下文切换 Context Switches/sec 不要大于 10000 到 20000。我们知道，CPU 不可能为一个程序长时间地服务，那样的话，其余的程序都会等待，从而导致服务质量下降。因此，CPU 采取的办法是在一个很短暂的时间内为这个程序服务，在下一个短暂时间片内为另外一个服务。这种方法有点类似众多围棋爱好者挑战一个九段高手的车轮战，使得在较短的时间内就可以满足很多爱好者与高手过招的要求。但

是，车轮战对高手的考验也是很严峻的，他需要记住自己和每一个爱好者当前局势的概要情况，如果爱好者过多，则胜率就会下降。对于 CPU 也是同样的道理，在切换程序进行服务的时候，CPU 需要保存当前为程序工作的状态以备下一次轮到它时使用。如果这种保存状态过多过于频繁，大量的时间会浪费在此上，导致性能下降。

（4）System 系统对象下的处理器队列长度 Processor Queue Length 不能大于系统处理器数目+1。操作系统会维护一个等待 CPU 处理的线程队列，如果在这个队列中，数目超过前面所说的值，将会出现处理器阻塞，导致 CPU 性能下降。在 Process 性能对象的% Processor Time 很高时一般都可能伴随有处理器阻塞。但是，若二者出现反比的关系，则需要查找处理器阻塞的原因。

6.6.2　有关多 CPU 与多核 CPU 的性能计数器

目前，越来越多的服务器采用了多个 CPU、多核 CPU 或者两者的结合作为中央处理器，因此性能也越来越强大。在这样的服务器上，读者可以通过任务管理器中的性能标签，了解到以每核为单位画出的 CPU 使用性能曲线。如图 6-30 显示的是一台安装了 Windows Server 2008，拥有 32G 内存，2 颗 4 核 CPU 服务器（在图右上方可以看到有 2×4=8 栏显示）运行任务管理器时，性能标签内显示的结果。

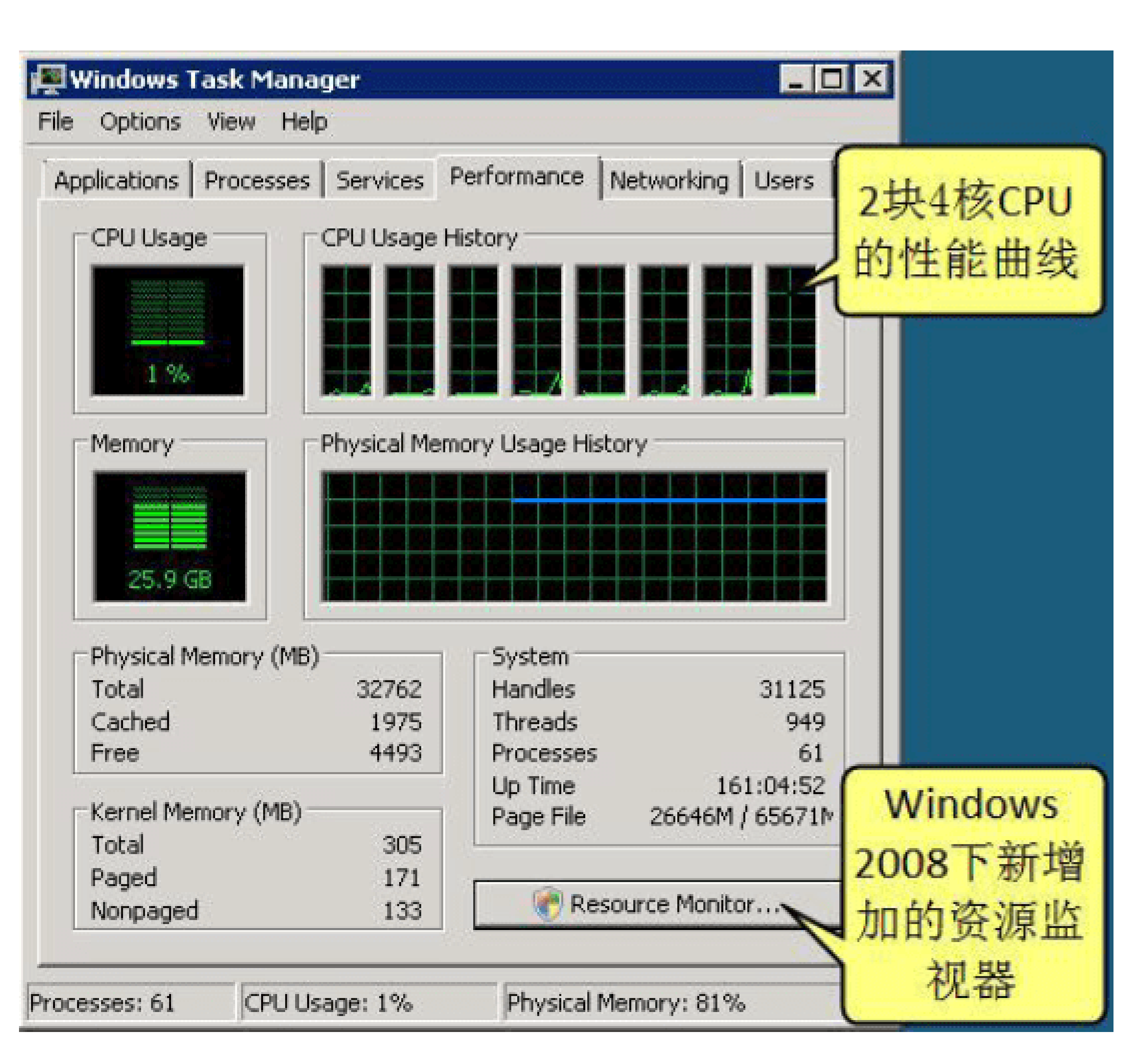


图 6-30　一个多 CPU 多核服务器上的实时性能曲线

那么，能否获得这些核心中某一个的性能计数器信息呢？

1．为某个CPU核心选择计数器

在图 6-30 所示的服务器上运行性能监视器（Perfmon.exe），添加 Processor 性能对象，可以发现，在下边（由于 Windows 2008 下界面有所不同，从右边变为下边）的 CPU 实例选择中，列出了所有核心（一共是 2×4=8 个，编号从 0 开始至 7 截止），而不仅仅是物理 CPU 的个数，如图 6-31 所示。

如果我们需要关注某个 CPU 核心的使用情况，选择编号，单击 Add（添加）按钮即可。

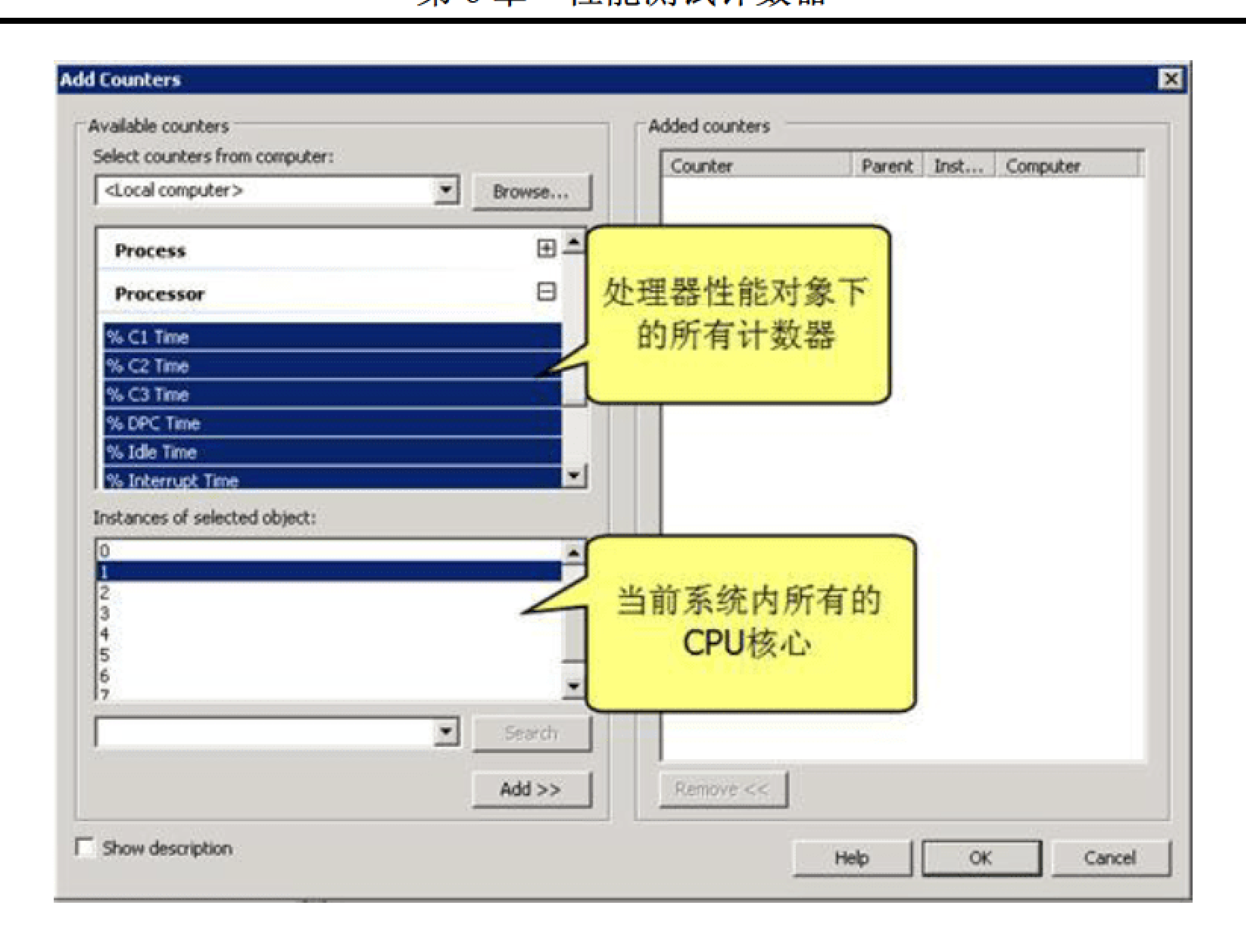


图 6-31　处理器性能对象列出了系统所有核心

从这里也可以发现，一块多核 CPU 与多个 CPU 在性能计数器方面并未被区分对待。

2. CPU时间超过100的问题

对于特定的一个进程 P 来说，选择 Process（进程）性能对象中的%Processor Time，即处理器时间计数器，实际数值可以超过 100。这样的情况会发生在多核心服务器上。其最大值可以达到 100N，其中 N 是所有 CPU 核心的个数。由于多个 CPU 核心都可以完全为一个进程服务，在此时如果进行统计，该进程耗费的处理器时间会将所有核心的处理器时间相加，造成百分比超过 100。同样，对于 Process（_Total）表示所有进程所耗费的处理器时间，最大值也是 100N。

而对于 Processor（处理器）性能对象来说，情况则有所不同。如果服务器中包含多个 CPU，每个 CPU 核心都会有自己的 Processor 性能对象数据，正如图 6-30 所示的那样。它们当中所有的计数器数值最大值都只可能是 100。另外，对于所有核心 Processor（_Total）统计的\%Processor Time 也会小于等于 100，因为它是多个 CPU 核心的平均值。

【注意不要被数据迷惑】

在实际工作中，如果看到处理器时间经常超过 100，那么就能知道服务器是否是多核的。前文所述的两点数值需要、区分也比较明确，因此我们不要被测试所得的数据迷惑。

6.7　磁盘性能分析

对于 Web 应用来说，有了计算能力强的 CPU，足够的内存还不够，还需要读写速度快，吞吐量大的磁盘做配合。因为 Web 应用大部分是网页程序加数据库的组合，磁盘性能出现瓶颈，将直接影响到用户请求网页、读取、保存数据的响应时间。

6.7.1　磁盘性能相关计数器

针对磁盘的性能计数器主要处于 LogicalDisk 和 PhysicalDisk 等两个性能对象中，一般需考察如下几种：

- LogicalDisk 性能对象下的%Free Space（空余空间）计数器。它表示当前没有占用的磁盘占总逻辑磁盘总容量的百分比。如果小于 15，则说明磁盘空间不足，需要考虑删除无用文件、增加磁盘等措施，否则会导致磁盘性能下降。
- PhysicalDisk 性能对象下的%Idle Time（空闲时间）计数器。它表示在当前观察时间间隔内，磁盘空闲的时间。如果数值小于 20，则说明磁盘工作状态较重，需要考虑替换成读写速度更快（比如转速更快、寻址更快、接口类型更快等）的磁盘。
- PhysicalDisk 性能对象下的 Avg.Disk Sec/Read（平均读取时间）计数器。它代表了每次从磁盘读取数据所花费的秒数。如果这个数值大于 25 毫秒（1 秒=1000 毫秒），则说明磁盘读取有所延迟。对于一些关键性的应用，比如 Web 应用中的数据库服务器，最好不要大于 10 毫秒。
- PhysicalDisk 性能对象下的 Avg. Disk Sec/Write（平均写入时间）计数器。它代表了每次向磁盘写入数据所花费的秒数。判断该项性能是否较好的标准，与前文平均读取时间的判断标准一致，也是 25 毫秒。
- PhysicalDisk 性能对象下的 Avg. Disk Queue Length（平均磁盘队列长度）计数器。它指出当前有多少 I/O 操作在等待磁盘进行处理。如果数值大于物理磁盘盘片数量+2，则会产生程度不同的延迟。
- PhysicalDisk 性能对象下的 Avg.Disk Byte/Transfer 计数器。它代表了在写入或读取操作时从磁盘上传送或传出字节的平均数。由于 Web 应用大部分是用户对于信息的获取，因此磁盘的读取操作要远大于磁盘的写操作。在合适的情况下（比如新闻、内容类网站）我们也可以用 Avg.Disk Byte/Read 计数器来代替。

除了以上的性能计数器，在实际工作中还要注意磁盘与内存问题的区分。

【区分磁盘与内存性能问题】

如果 Avg.Disk Byte/Transfer 数值较低，但%Idle Time 数值却很低，Average Disk Queue Length 的值又很高，则很大程度上说明磁盘性能存在瓶颈：磁盘空闲时间很少，忙于处理读写操作，但等待的工作却很多。但如果 Average Disk Queue Length 较高，Avg.Disk Byte/Transfer 也比较高，则不能说明磁盘的问题，很可能是内存不足造成的数据排队等待进入磁盘现象。

6.7.2　与其他性能对象的综合考虑

实际上，磁盘 I/O 受到内存、CPU 性能对象的影响比较大，而且这种影响是相互的。在对磁盘 I/O 进行分析时，将其他性能对象考虑进来是很常见的做法。比如，利用 Memory 性能计数器下的 Cache Bytes（缓存字节数）计数器。它代表了系统使用多少内存用于文件缓存。如果数值大于 200M 字节，则表明磁盘存在瓶颈。

6.8　网络性能分析

对于 Web 应用来说，用户的请求总是要通过服务器的网卡才能得到反馈的。随着 Web 应用功能的不断丰富、界面越来越酷炫，需要网卡的数据吞吐量也越来越大。因此，网络成为系统性能瓶颈的情况很有可能存在。

常用的网络性能分析计数器有如下两个：

（1）Network Interface 性能对象下的 Bytes Total/sec 计数器。它标明了当前网卡发送、接收字节的速率。如果数值大于网卡标称数值的 70%，则说明网卡性能存在瓶颈，需要更换速度更快的网卡，或者采取措施减少网络无关流量。

目前，Web 应用服务器采用的至少都是 1000Mbps 的网卡，则网卡的标称数值约为 1000Mbps=1000000kbps=125Mbytes/sec。当前 Bytes Total/Sec 计数器数值如果大于 125×70%=87MB/sec，则说明网卡负担过重。

（2）Network Interface 性能对象下的 Output Queue Length 计数器。和之前介绍的诸多队列计数器类似，它表明了等待网卡处理的队列中包的长度。如果大于 2，则说明网卡工作负担较重。

除了服务器上的网卡之外，服务器之间、机房接入的网络也可能产生很大的瓶颈，不过这已经超出了服务器系统的范围，也没有性能对象与此对应，因此，并不是性能测试工程师的职责范围，需要专门的网络分析人员、网络管理人员甚至网络安全人员的全力参与。

6.9　应用服务器性能简要分析

前文讲述了操作系统和服务器的一些性能分析方法。对于一个主流的 Web 应用来说，光有它们的支持还不够，Web 应用程序往往需要运行在一些应用服务器上，比如微软公司的 IIS，Oracle（BEA）公司的 Weblogic 等。本节将通过这两个应用服务器的例子，说明对它们的性能分析方法。

6.9.1　IIS 应用服务器性能分析

IIS 全称是 Internet Information Service，为 Windows Server 2003 和 2008 系统自带（对于后者一般需要自行安装），可以为用户提供 WWW 服务、FTP 服务、SMTP 服务等。对于大中型网站来说，都只是利用它的 WWW 服务。我们以 Windows 2003 系统中自带的 IIS 5.1 为例，根据需要也会提及新版本 IIS 6 和 IIS 7，它们虽然在总体架构上变化不小，但是在性能计数器方面是基本一样的。

在了解 IIS 相关的性能计数器之前，有必要对 IIS 相关的进程和服务名称有所熟悉。在 Web 应用所在的服务器上，它们这样的进程在任务管理器中占用大量 CPU 时间是比较正常的。熟悉了这一点，测试工程师不至于产生不必要的疑惑。这些进程和服务分别如下。

❑ Inetinfo.exe：默认位置在 C:\WINDOWS\system32\inetsrv 下，在 IIS 5.X 中负责响

应用户请求。

- Http.sys：运行在 Kernel 模式下（Windows 的不同模式请见图 6-28），在 IIS 6 和 IIS 7 中负责响应用户请求。
- Aspnet_wp.exe：在 IIS 5.X 中处理 ASP.NET 服务，在服务器的任务管理器中只会列出一个。
- W3wp.exe：在 IIS 6 和 IIS 7 中负责具体的某个 Web 应用，可以有多个列在任务管理器中。

6.9.2　IIS 相关性能计数器

由于 IIS 是微软出品的 Web 应用服务器，因此绝大部分使用 IIS 的网站都采用了 ASP、ASP.NET 技术。我们需要了解和它们相关的性能计数器。

1. Active Server Page性能对象

主要关注如下若干计数器：

- Requests Time Out（请求超时数量）；
- Requests Rejected（拒绝请求数量）；
- Requests Failed Total（请求失败总数）；
- Requests Total（请求总数）；
- Requests/Sec（每秒请求数量）；
- Requests Queued（已排队的请求数量）；
- Request Wait Time（请求等待时间）；
- Request Execution Time（请求执行时间）。

在这些计数器中，如果“已排队的请求数量”较大，“请求等待时间”较长，说明 IIS 处理用户请求出现瓶颈。

通过“请求失败总数”与“请求总数”的对比，可以得知用户成功访问网页的比例。具体的请求失败可以通过请求超时、请求被拒绝、未授权的请求等计数器来获得。

2. ASP.NET性能对象

与 ASP 性能对象类似，ASP.NET 性能对象也主要关注请求的诸多方面。值得注意的是，ASP.NET 并不是一个性能对象，而是分成多个，有 Applications、具体 ASP.NET 版本等。

3. .NET性能对象

由于 ASP.NET 运行需要.NET CLR（公共语言运行库）的支持，因此查看以.NET 开头的若干性能对象也有助于多了解当前 Web 应用对于系统性能的影响。

4. Web Service性能对象

如果 Web 应用对外提供了 Web Service，可以考虑查看如下等几个计数器：

- Connection Attemps/Sec（每秒 Web Service 尝试连接数量）；

- ❑ Maximum Connections（Web Service 的最大连接数）；
- ❑ Anonymous Users/Sec（每秒使用 Web Service 的匿名用户数量）。

通过这些计数器，我们可以得知 Web Service 当前的真正负荷，工作质量如何（可以利用错误与总数的比例表示）。

6.9.3　Weblogic 性能信息的直观获得

与微软的.NET 开发平台对应的是 Java 平台，在这个平台中常用的 J2EE 应用服务器包括 Oracle（BEA）的 Weblogic、IBM 的 WebSphere 以及免费的 Tomcat 等。在这里我们以应用较广泛的 Weblogic 8.1 为例，对性能测试中应该考虑的主要计数器做一下介绍。

【Weblogic 性能信息的直观获得】

Web 应用通过 Weblogic 部署后，都可以通过浏览器的方式进行管理。在这里以 Weblogic 8.1.5 版本为例，打开管理界面，选择当前服务器，在右边的内容栏中依次选择 Monitor 和 Performance 链接，就能发现当前 Web 应用的性能信息，如图 6-32 所示。最新版本的 Weblogic 管理界面也基本类似，读者可以查阅相关的管理员手册。

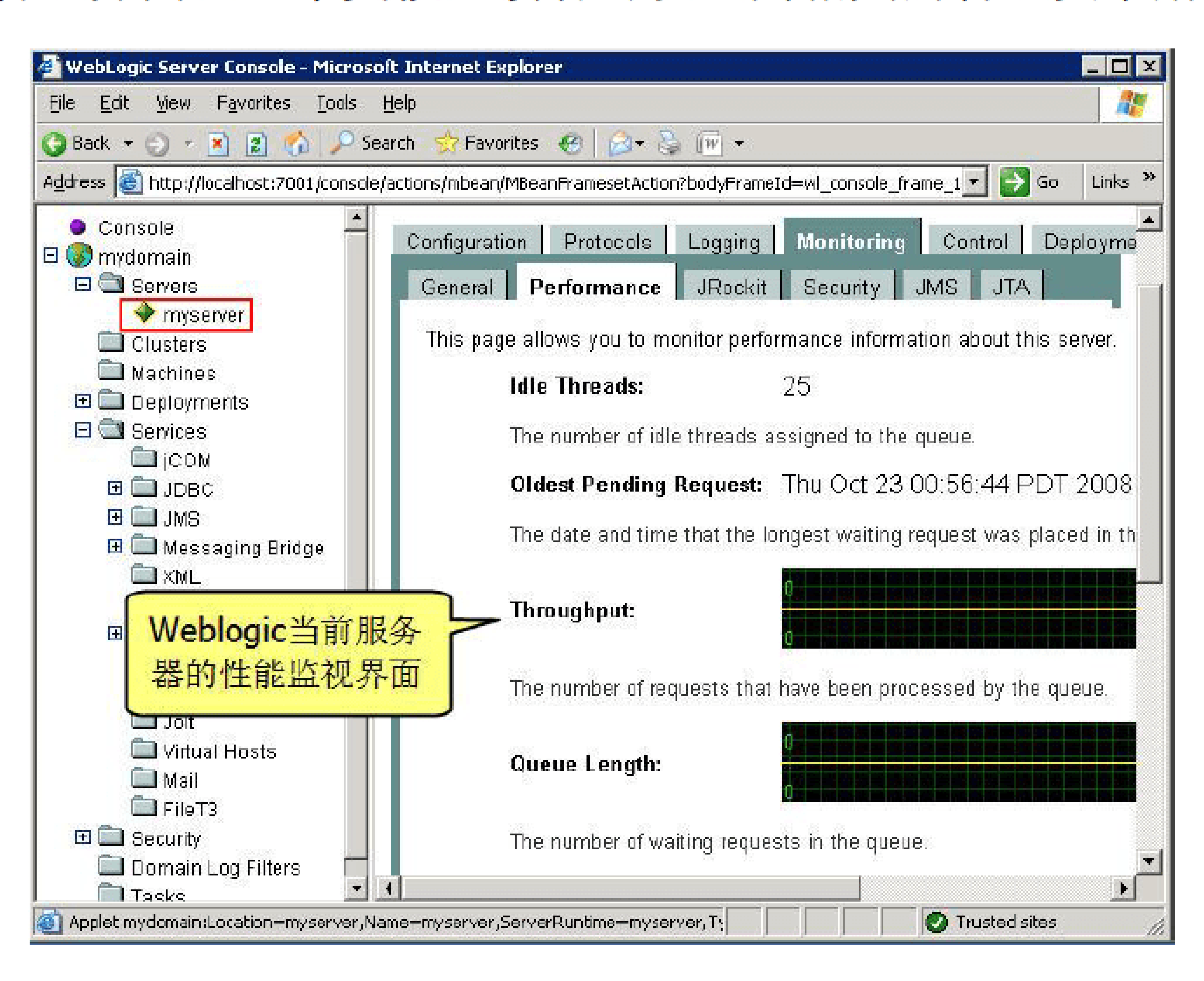


图 6-32　在 Weblogic 管理页面中发现性能信息

Weblogic 并没有将自己的性能计数器提供给 Windows 系统，这可能出于产品要跨平台的缘故，通过自定义的界面获得性能信息，可以省去为不同系统开发性能监视功能的成本。

6.9.4　Weblogic 相关性能计数器说明

由于具有的“承上启下”“中间人”作用，Weblogic 等 Java 平台应用服务器的性能与如下一些因素相关：

- ❑ 所在操作系统的软硬件性能；
- ❑ 应用服务器连接数据库系统的性能；
- ❑ Java 虚拟机的性能；

- ❑ 网络传输性能；
- ❑ Web 应用相关性能，比如：
  - ➢ EJB：企业 JavaBean。
  - ➢ JSP and Servlet：Web 应用的表现层，网页及相关部分。
  - ➢ JMS：Java 消息服务。
  - ➢ JDBC：Java 连接数据库的桥梁。

实际工作中，需要性能测试工程师额外关注的方面如表 6-6 所示。

表 6-6　Weblogic常用性能计数器列表

性 能 对 象	计数器名称	计数器描述
JVM Java 虚拟机	Heap Size	JVM 堆大小，该计数器的值是一个实时值
	Free Heap	JVM 可用堆大小，该计数器是的值是一个实时值
	Max/Min Heap	JVM 可用堆最大/最小值
JDBC Connection Pool JDBC 连接池	Connections Total Count	总的 JDBC 连接数
	Waiting For Connection Current Count	等待的连接数量。如果该计数器的值持续比较大，需要考虑增加应用服务器的 JDBC 连接池设置
	Max Capacity	JDBC 连接池总容量
	Active Connections Current Count	当前活跃的 JDBC 连接数。通过该计数器的值，可以知道 JDBC 连接的利用率是否合理
Execute Queue 执行队列	Thread Count	线程数量。一般稍微大于 CPU 数量比较合适
	Execute Thread Current Idle Count	空闲的线程数量
	Pending Request Oldest Time	队列请求的最久时间
	Serviced Request Total Count	已处理的请求总数
	Pending Request Current Count	挂起请求的数量

对于其他的 J2EE 应用服务器来说，为了开发、调试的方便，都会提供类似的性能计数器。我们在测试之前需要仔细阅读它们的帮助文档。

值得一提的是，应用服务器的性能受到操作系统软硬件各部分性能、运行于其上的 Web 应用程序代码性能的双向制约，因此，在实际工作中，要从更广的方面考虑性能问题。

6.10　数据库性能简要分析

数据库在 Web 应用中起到了类似“后方根据地”的作用，这是因为：绝大部分的 Web 应用数据是存放在数据库中的；绝大部分的用户操作记录也是存放在数据库中的。大部分的 Web 应用代码都是和数据库的读取、写入、更新、删除等操作直接或间接相关。因此，数据库性能好坏对整个 Web 应用的表现影响很大，而为了有的放矢，在测试前对数据库性能进行一个基本的判断也是很有必要的。

6.10.1　业内常见的数据库产品

目前，网站使用的常见数据库与开发平台有如下几种配搭：

- ❑ .NET/ASP 开发平台 + SQL Server（版本一般使用 2000，2005 和 2008 这几种）；
- ❑ PHP 开发平台 + MySQL（截止目前最新版本为 5.0）；
- ❑ Java 开发平台 + Oracle（版本有 8i，9i，10g，11g 等）。

还有其他的一些数据库产品，因为不占主流就不列出了。这 3 种数据库分别是微软公司、Sun 公司和 Oracle 公司的产品。

6.10.2　数据库性能问题对应的性能计数器

数据库系统是比较复杂的，有时候问题需要积累一段时间才能发现。因此在本节，本书将计数器与性能问题、解决对策联系起来。由于篇幅所限，本书不可能对所有的计数器都进行解释，仅以微软公司的 SQL Server 为例，如表 6-7 所示。这些计数器都是 SQL Server 自带的，并且可以通过 Windows 系统下的性能监视器（Perfmon.exe）加载。

表 6-7　SQL Server 2005 数据库常用性能计数器

性 能 对 象	计数器名称	含义及与实际问题的联系
SQL Server Access Methods	Page Splits/sec	每秒钟分页次数。 超过 100 可以视为较大，将使得磁盘 I/O 变大，搜索变慢，性能变差。 可以通过增大填充因子减少
SQLServer: SQL Statistics	Batch Requests/Sec	每秒钟批次请求数量。是描述 SQLServer 是否忙碌的标准之一。 超过 1000 到 5000 可以视为较大。 会影响网卡性能，可通过增加网卡或者更换更高级别网卡解决
SQLServer: SQL Statistics	SQL Compilations/Sec	每秒钟 SQL 语句编译数量。 超过 100 可以视为较大。会影响查询性能。 可以通过增加预先编译好的存储过程来提高速度
SQL Server General Statistics	User Connections	当前所有用户连接数。 可以用来进行数据同比等，比如描述一天中数据库用户的变化规律。 通过增加硬件提高用户连接数
SQL Server Locks	Number of Deadlocks/sec	每秒死锁的数量。 需要 SQL Profiler 来进一步分析代码中死锁可能发生的原因
SQL Server Locks	Average Wait Time (ms)	平均锁等待时间。 时间较长可导致用户提交数据更改时需要长时间才能完成。 对于一百万行数据的表，超过 500ms 的锁等待时间可以视为较大

续表

性 能 对 象	计数器名称	含义及与实际问题的联系
SQL Server Access Methods	Full Scans/sec	每秒全表遍历次数。 对于查询，全表遍历一般是比索引查询更费时间的。 对表应用合适的索引可以减少全表遍历
SQLServer:Memory Manager	Total Server Memory (KB)	SQL Server 使用的全部内存。 尽量将数据库服务器的内存交给 SQL Server 使用

以上只简要列出了一些比较重要的 SQL Server 计数器及相关的实际问题。对于其他的主流数据库，它们的功能大同小异，计数器也是如此，读者在工作中可以举一反三。

数据库系统的性能优化一般都是在一轮测试结束后进行，因此本书将其放置在末尾的第 17 章。当然，数据库本身就是计算机学科中的一个重要分支，读者若打算深入地研究，需要阅读更多的书籍，并在工作中积累丰富的实践经验。

6.11　本 章 小 结

性能测试的原始结果就是有关于系统性能或者 Web 应用性能的数值。这些数值可以通过操作系统提供的诸多性能计数器获得。本章首先介绍了 Windows、Mac OS X 和 Linux 下获得性能计数器的方法，并以 Windows 系统为例，讲解了其下很好的性能监视器程序（Perfmon.exe）的基本使用方法。

在本章的后面部分，进一步介绍了如下几种重要的性能对象分析方法：

- ❑ 内存；
- ❑ CPU；
- ❑ 磁盘；
- ❑ 网卡。

它们都有相应的多种计数器供我们单独或者综合地判断是否出现性能问题。每个性能计数器都有一定的阈值（即说明出现问题的边界值），不过，这个阈值并不一定是固定的，需要根据实际情况进行调整。

对于 Web 应用来讲，只对上述若干部分进行性能分析是不够的，读者还需要考虑应用服务器和数据库服务器的性能。这是本章第 3 部分讲述的知识，并分别以微软公司的 IIS、SQL Server 和 Oracle 公司的 Weblogic 产品做了入门讲解。

本章提供的分析方法可以说是一种定性的分析方法，只能够指出大致的问题所在。在实际的性能测试工作中，能够起到基本作用，但并不是全部。每一个 Web 应用都有自己的特点，性能测试工程师需要在深入了解当前被测试应用的系统组成、功能实现方法，采用的应用服务器、数据库服务器的特点之后，在采用本章的分析方法之外，制定出更具体、更符合实际应用场景的方法，才能达到较好的性能测试效果。

第 3 篇　使用 LoadRunner 进行 Web 应用性能测试

第 7 章　LoadRunner 的基本使用

第 6 章中，学习了各个操作系统、应用服务器和数据库的性能计数器，了解到性能好坏可以通过这些计数器的数值体现出来。在实际的性能测试工作中，我们也需要经常获得这些数值。但是，有两个问题就会摆在面前：

- 如何模拟大量用户对 Web 应用的访问？在第 5 章介绍过很多性能测试方法都需要模拟这样的场景。
- 如何自动化地获取性能计数器的数值？如果依靠人工，那将是非常耗费时间的过程，并且容易出现错误。

要解决以上两个问题，都需要自动化测试工具的参与。俗话说：“工欲善其事，必先利其器”。有一个好的性能测试工具软件，会极大促进解决上述问题的效率和质量。小白因此也明白了第 5 章经理所布置的最后一项任务：测试工具软件选型的意义。本章我们就来帮助小白完成如下两件事情：

- 首先需要选择一个比较好的性能测试工具软件。
- 其次熟悉这个工具软件的基本操作。

掌握一种常用的性能测试工具软件，对于日后遇到其他软件上手也会比较快，可以“举一反三”：因为所有的性能测试软件都需要面临和解决本章开始提出的两个问题。

7.1　测试工具软件的选择

在目前市场上，有很多现成的商业或者免费测试工具软件可供挑选，性能测试软件在其中占有很大的一部分。其实，测试工具不一定非要从市场上购买，也可以自行编写。那么自创和购买各有什么优点呢？

7.1.1　自行编写与购买测试工具的比较

对于小白这样并没有多少软件选型经验的初级测试工程师来说，我们不妨为他出些主意。软件离日常生活有点远，我们可以用新衣服的选择来谈这个问题。假设小白需要购买一些衣服，有两种方式：自己做（或者找裁缝做）还是在商场购买呢？这两者的优缺点各是什么？

我们把两种方法的优缺点都列在一张纸上，有如下的结果。

1．自己做或者找裁缝做的方式

- 优点

  - 如果自己或者裁缝水平高的话，可以做到完全合身。
  - 成本可以选择：可以选择不同的布料进行加工，可贵可便宜，控制起来方便。
  - 过程有时候会有从无到有的成就感。
- 缺点
  - 自己不会做衣服的情况占 99%（不排除有些读者是家务能手），好的裁缝需要寻找。
  - 需要较多的时间。无论是自己做还是裁缝做，总要量体裁衣，然后试穿修改等，往往不能立刻拿到新衣服。

2．在商场中购买成衣的方式

- 优点
  - 到商场根据自身的号码，很快就能买到比较适合的衣服。相对上一种方式可以节省时间。
  - 不需要自己对布料、裁衣手艺有任何深入的了解。
- 缺点
  - 如果小白的身材比较特殊，在商场挑到合适的衣服可能比较困难。
  - 成本在很多情况下会比同样的衣料自己或者找裁缝做要贵，毕竟多了商家的利润、流通环节的利润等。

其实，购买测试工具软件与购买衣服的选择是类似的，不过测试工具软件需要更多的学习成本（使用者花费在学习工具上的时间）和维护成本（在使用当中，需要投入一些机器来运行软件、需要投入时间对软件进行管理等）。另外，测试工具软件还需要考虑长期使用还是短期使用的问题，只为当前项目购买一个比较昂贵的商业软件需要考虑一定的成本。同样，自行开发的软件测试工具在不同时间都能满足不同项目（也可以是一个项目和它的后续版本）的需求也比较困难。

综上所述，并考虑到其他的一些方面，表 7-1 列出了两种方式的对比。

表 7-1　购买与自行编写测试工具软件的优缺点比较

自行编写方式	购　买　方　式
最接近工作实际，从本项目需求出发，多余功能很少，同时，对当前项目的测试比较完全	商业软件能够满足通用功能 本项目需要的额外功能无法提供，较难扩展，而且很可能浪费软件提供的部分功能
由于自行开发，开发者在公司内部，易于学习和使用	商业开发，客户的反馈无法快速变为现实。 功能复杂，需要学习时间较长，并且强烈依赖于工具的易用性以及所提供帮助文档的可读性、正确性
一般来说，稳定性和可靠性未经充分验证。如果出现问题，需要判断是该工具的问题，还是被测试系统真正的 Bug	一般是被完整测试过的商业软件，而且稳定性、可靠性方面有开发商的实力、信誉以及利润需求的保证
开发成本高，特别是需要时间进行开发	基本不需要开发成本，购买软件只需要购买计划与等待预算批准等少量时间
购买成本低，开发人员工资已经包含了大部分成本	一般需要很大的购买费用，增加除工资外的支出
可以成为公司的财产，并且通过时间和使用的积累，也有可能发展成为商业产品投放市场 可以形成独有的测试工具软件体系和文化	采购自不同公司的不同产品很难集成，需要不同的人员负责，不利于工作内容和知识的共享

续表

自行编写方式	购 买 方 式
在开发过程中培养了员工的能力	商业软件的使用过程对培养员工能力提高不大
知识产权的要求。有些情况下，测试工具在项目中的使用是受到限制的	在允许的情况下，使用完全没有限制，没有产权方面会出现的纠纷

【实际的选择方法】

一般地说，自行编写测试工具适合与被测软件、Web 应用紧密相关，特殊要求的场合；而购买商业软件方式适合测试软件、Web 应用的通用功能和需求。在实际工作中，除非特别考虑（比如成本方面、知识产权方面、发展战略、或者就是市面上还没有这类商用工具等），都采取两者相结合的方法：通用需求测试使用商业软件，特殊需求使用自行编写的测试工具。

在明白购买还是自行编写测试软件的判断标准之后，小白还需要了解自己所在的网站需提供哪些功能，市场上的商业软件又能提供哪些方面的性能测试，才能进一步给经理提出自己的建议。因此，下面我们将简要介绍从哪些方面来考察常用商业性能测试软件的功能。

7.1.2　常用的性能测试工具软件

要在短时间内了解一个市面上的测试软件，最好的方法就是：

- ❑ 访问官方网站，了解软件运行的基本配置要求。
- ❑ 在官方网站下载试用版本，进行基本的实际操作，验证该软件是否符合要求。

【软件的试用版本】

所谓试用版本，英文名称一般是 Evaluation Copy 或者 Evaluation Edition 等，是软件开发公司为了让用户能够了解其功能所提供的。试用版本一般在该公司网站上都能免费下载到（有的时候需要注册成用户，便于公司了解客户构成等信息）。试用版本大多在功能上与正式版本相差无几，但是在使用期限上有所限制，一般是 30 天，也有 10 天、60 天甚至 180 天（比如 Windows 的某些试用版本等）的。

常见的专业性能测试软件有如下几种。

- ❑ LoadRunner：业内最常用的性能测试软件之一。由 Mercury 公司开发，后来被惠普公司收购，成为它产品线上的一员。对主流技术和场景支持全面，比如 J2EE、.NET 等开发平台、Web 应用、客户端/服务器应用、CRM 应用等等。官方网站：https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-126-17^8_4000_313__
- ❑ SilkPerformer：曾经非常辉煌的 Borland 公司生产的产品，对应用的支持也比较全面。官方网站：http://www.borland.com/us/products/silk/silkperformer/index.html。
- ❑ QALoad：Compuware 公司的产品。官方网站：http://www.compuware.com/products/qacenter/qaload.htm。
- ❑ Rational Performance Tester：蓝色巨人 IBM 公司的性能测试产品，是 Rational 系列软件的一部分。官方网站：http://www-01.ibm.com/software/awdtools/tester/performance/features/index.html。

- OpenSTA：Windows 平台下开放源代码的免费性能测试工具。官方网站：http://www.opensta.org/。
- Forecast：英国 Facilita 公司开发的性能测试软件，分为多个套件，每种满足一种平台或场景（比如 Forecast for Web、Forecast for Java 等），因此可不必购买支持无关平台的软件。官方网站：http://www.facilita.co.uk/web2/index.php?section=14。
- E-Load：原为 Empirix 公司开发的 e-Tester 套件中的一部分，专门进行 Web 性能测试的软件。该套件在 2008 年上半年被 Oracle 公司收购，将其作为后者杀入软件测试领域的一个产品。在不久的将来，它或许能获得与惠普公司的 LoadRunner，IBM 公司的 Rational Performance Tester 同样的地位。官方网站：http://www.scl.com/products/empirix/testing/e-load。

本书的附录 A 还列举了其他一些较有特色的性能测试软件。

7.1.3　性能测试工具软件的评估

小白在得到当前市场上常见的性能测试软件列表后，下一步就要进行简要的评估过程了。这个过程可以分为 4 步。

- 列出公司网站常见的测试场景、测试需求。
- 在官方网站上查询、与软件代理商咨询相关功能能否实现，针对公司网站，列出每个软件可实现的功能。全部覆盖者取胜。
- 考察性能测试软件用户是否较多，技术支持是否及时；可提供或者可搜索到的培训资料是否丰富。这与挑选饭馆看其中就餐的客人多不多有异曲同工之妙。
- 成本的考虑。小白负责提建议，只需要将价位列出来即可，其他的事情还需要经理与技术总监进行沟通和抉择。在这方面，需要知道大部分性能测试软件（或者大部分商业软件）都是根据支持的并发用户（在这里，就是同一时刻访问 Web 应用的用户数）数来定价的，并发用户数量越多，软件就越贵，因此要合理估计网站建成后实际的并发用户数量。一般来说，数值 1000 对于一般的中型网站，已经是相当大了。更大型的网站，可以选择无限制的版本。

从功能上考察性能测试软件的要点

由于大部分网站的功能都类似，因此对这些功能的测试需求也类似。对于从功能上评估一个测试软件，可以从表 7-2 中的几个方面进行考虑。

表 7-2　从功能上考察性能测试软件的几个方面

类　　别	具 体 内 容
软件、硬件需求	该软件能否支持被测 Web 应用所在的系统、所采用的技术平台（比如 Windows、Java 技术、SQL Server 这样的数据库）
Web 应用协议	该软件能否对被测 Web 应用所采用的协议进行分析。比如网站必备的 HTTP 协议和有些网站需要的 HTTPS 协议等
计数器支持	该软件能否自动获取服务器、应用服务器和数据库服务器的各项性能计数器指标。这对于自动化测试是非常重要的，也是我们选择测试软件的原因之一

续表

类　　别	具 体 内 容
Web 应用相关技术	该软件能否支持一些 Web 应用的特殊环境，比如负载均衡、服务器代理、防火墙以及 NAT 技术等
可编程能力	该软件能否支持自定义性能测试场景的编程，编程所用脚本是否简单易学

7.1.4　小白的最终选择

在 7.1.2 节中列出的主流性能测试软件中，目前市场占有率最高的是 HP 公司的 LoadRunner 和 IBM 公司的 Rational Performance Tester，其中前者占据整个市场份额的 60%～77%不等（据不同专业公司调查得到的数据）。从数字可见，LoadRunner 在性能测试领域获得了较多用户的认同。

考虑到公司网站并没有很多特殊的、需要自行开发测试工具的场合，另外出于市场占有率高的原因，小白决定向经理建议购买 LoadRunner。市场占有率高有如下几个好处：

- 使用用户较多，说明它相对能适应不同用户的应用场景。
- 使用用户较多，间接说明用户的反馈就多，使用过程中软件出现的 Bug 就会被更多人发现，可以令公司及时地修正，从而至少在心理上增加测试软件的可靠性。
- 使用用户较多，也利于个人发展，因为职场上可以提供的相关职位较多，便于人才流动。

当然，第 3 点是出于小白自己的考虑。不过从公司方面着想，一个使用者多的软件，也意味着人才市场上可以找到的潜在员工较多。因此，在这一点上，选择它也是公司、员工双赢的事情。

最终，小白的建议得到了经理的批准，虽然离购买、投入实际使用还需要一段手续和财务上的时间，但他很快就可以得到真正开展性能测试的机会了。

在这一段等待的时间，小白决定提前熟悉 LoadRunner 这个软件。在 7.2 节，我们将和他一起下载软件的试用版，“试驾”一番。

7.2　LoadRunner 的下载与安装

LoadRunner 原来由 Mercury 公司开发，后来公司被惠普公司收购，因此它也就成为 HP 公司的产品。LoadRunner 最新的版本，截至 2008 年 10 月是 9.10。

LoadRunner 的试用版本可以使用 10 天，相对于其他软件是比较短的。由于公司的网站开发已经接近尾声，功能测试，系统测试也已经进展较多，小白希望在这 10 天中能够尽量熟悉基本的操作，从而在正式软件购买回来后就可以尽快地进行第一次自动性能测试，给开发部门和其他相关同事以性能方面的信心。

7.2.1　LoadRunner 的下载

要下载 LoadRunner，首先要成为 HP 网站的免费注册会员（称为 Passport，护照）。

注册网址为：https://h10078.www1.hp.com/cda/hpdc/display/main/register.jsp。

注册成功后，在网站的搜索文本框中输入 LoadRunner 就可以查询到试用版本（Evaluation version）的下载页面。打开后的下载页面如图 7-1 所示。

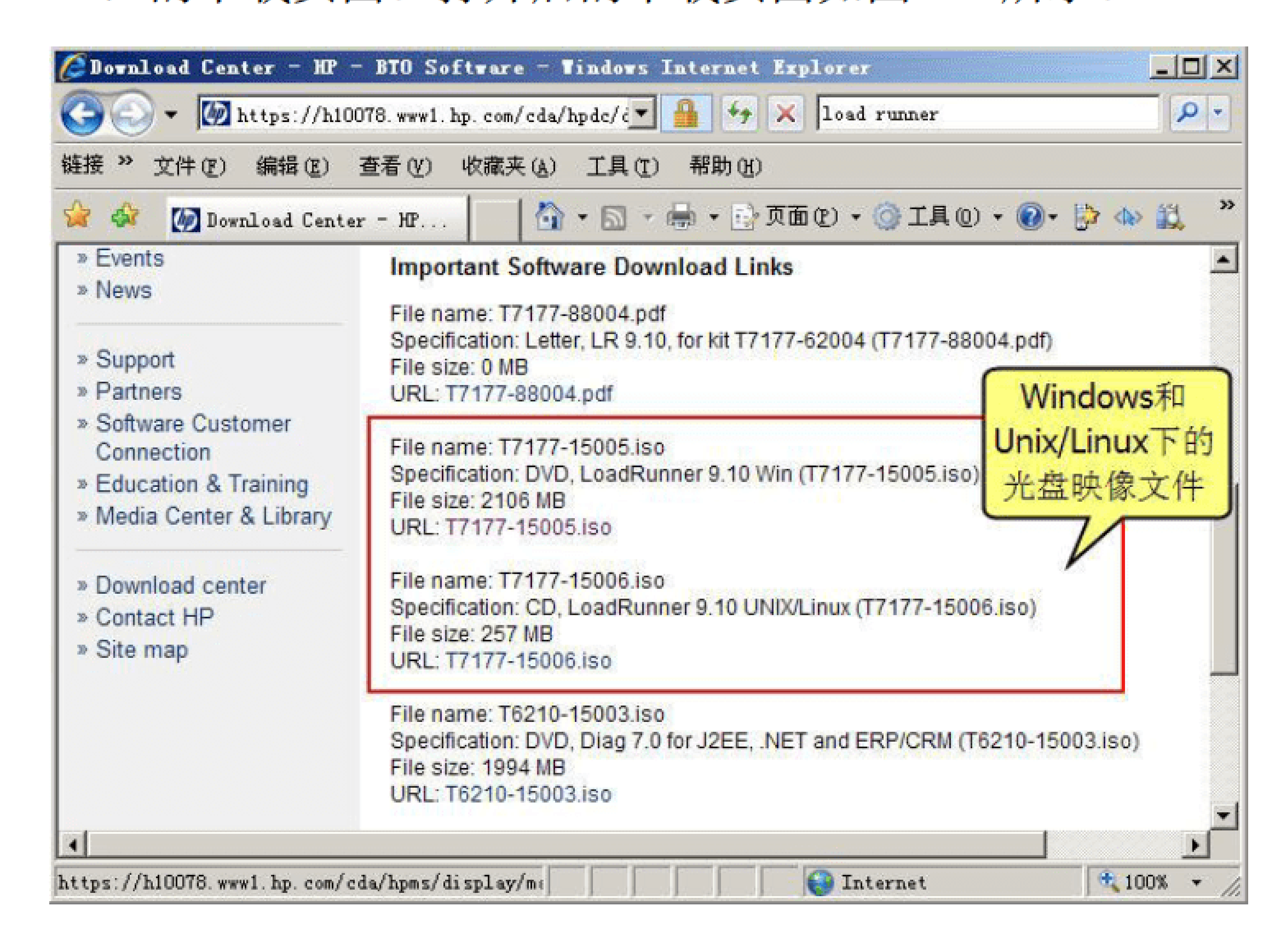


图 7-1　LoadRunner 试用版本的下载页面

注意，在图 7-1 中，有两种版本可供选择，分别是 Windows 版本和 Unix/Linux 版本。下载的文件格式为光盘映像文件，后缀名为 ISO。可以直接用虚拟光驱软件打开，或者直接刻录成光盘（大小为 2G 多，需要刻录 DVD）进行安装。下载完毕并加载到光驱后，单击 Setup.exe 即可进行安装。

7.2.2　LoadRunner 的安装

在本书中，为了方便，我们选择 LoadRunner 9.0 进行讲解，这是因为最新版本刚刚面世，在实际工作中能遇到的还是大量使用的 9.0 版；另外，在功能上，两者变化也不大。

LoadRunner 的安装界面很简单，与其他通用软件没有太多不同。首先出现的是软件版权屏幕，紧接着如图 7-2 所示的安装向导开始运行。在界面上，读者可以发现左边列出了安装的步骤如下。

（1）Welcome 步骤：欢迎屏幕。

（2）Setup Type 步骤：完全安装还是自定义安装。

（3）Confirmation 步骤：确认安装信息。

（4）Installation 步骤：具体复制文件、注册组件等的安装过程。

（5）Finish 步骤：结束安装的界面。

从以上 5 个步骤来看，LoadRunner 安装是比较简单的。

在图 7-3 中选择 I Agree（我同意）单选按钮，一直单击 Next 按钮，就可以成功地安装 LoadRunner 了，很简单。默认的安装位置是在系统盘（笔者电脑为 C 盘）Program Files 文件夹下新建立的名为 Mercury 的子文件夹。之所以该文件夹称为 Mercury 的原因，在本节开始已经谈过。

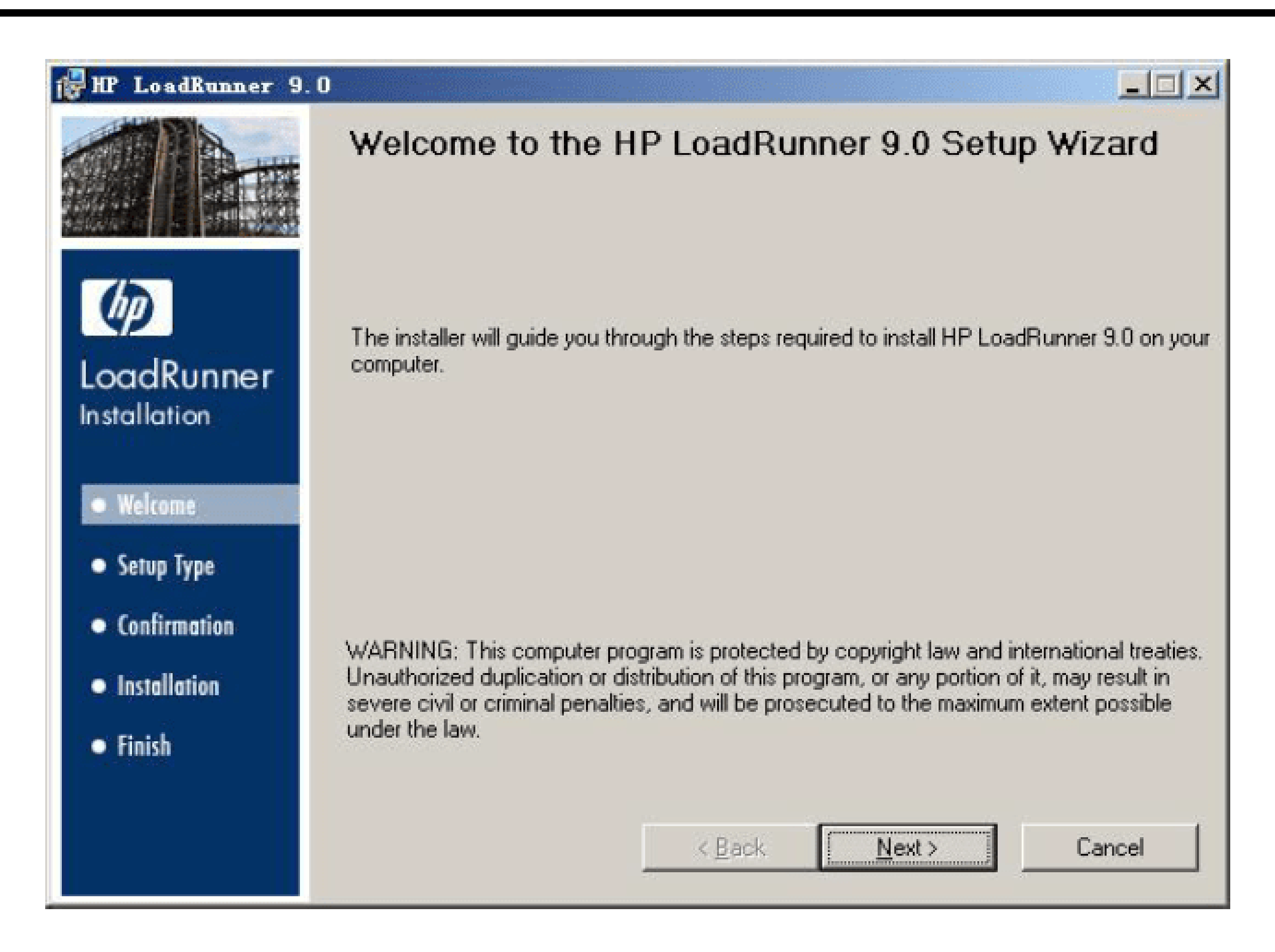


图 7-2　安装界面左边列出了安装过程的 5 个步骤



图 7-3　用户使用协议界面

【题外话：安装与 UI 测试】

作为一名测试工程师，保持一双发现 Bug 的慧眼是非常重要的。这种能力需要在使用任何软件的时候进行练习。对于 LoadRunner 的安装过程，我们也可以进行两类测试：安装测试和 UI 测试。安装测试验证软件可否正常安装，比如安装在不同的磁盘、安装在空间不足的磁盘，安装在不存在的目录等很多情况下能够顺利完成或者提示正确的错误信息。对于 UI 测试，我们需要验证所有安装界面的 UI 都正常，比如用户协议文字是否正确、图片能否显示、快捷键能够正确激活等。要知道，在实际工作中，单纯的性能测试工程师是比较少见的，他们往往都兼任其他的测试工作。通过平时的这些思考和积累，可以为我们的测试生涯提供更多经验。

安装完毕后，就可以进行 LoadRunner 的“试驾”了。

7.3　LoadRunner 入门

在系统任务栏上单击“开始”按钮，在弹出菜单中选择“程序”菜单下的 LoadRunner

程序组，单击 LoadRunner 程序项，就可以运行该软件，开始性能测试的旅程。

由于在 7.2.2 节中，小白下载的是试用版本，只有 10 天的使用期限，因此每次启动 LoadRunner 都会出现一个倒计时的窗口，如图 7-4 所示。在图中单击关闭按钮后才能进入主界面，如果是正式版本，这一个过程是没有的。

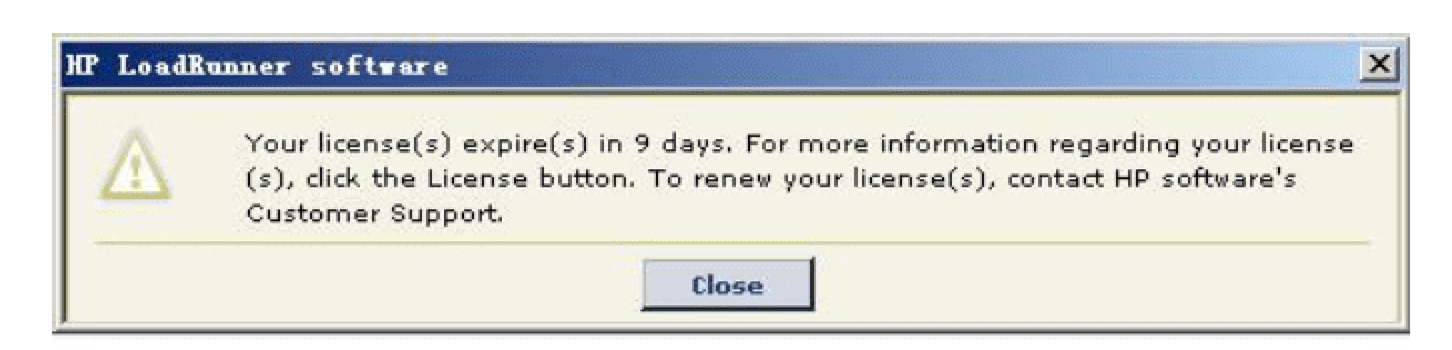


图 7-4　LoadRunner 试用版本的倒计时提示

7.3.1　LoadRunner 的导航窗口

紧跟图 7-4 出现的就是 LoadRunner 9.0 的导航界面，它包含若干个链接，为使用者的不同目的提供分类和切换界面，如图 7-5 所示。

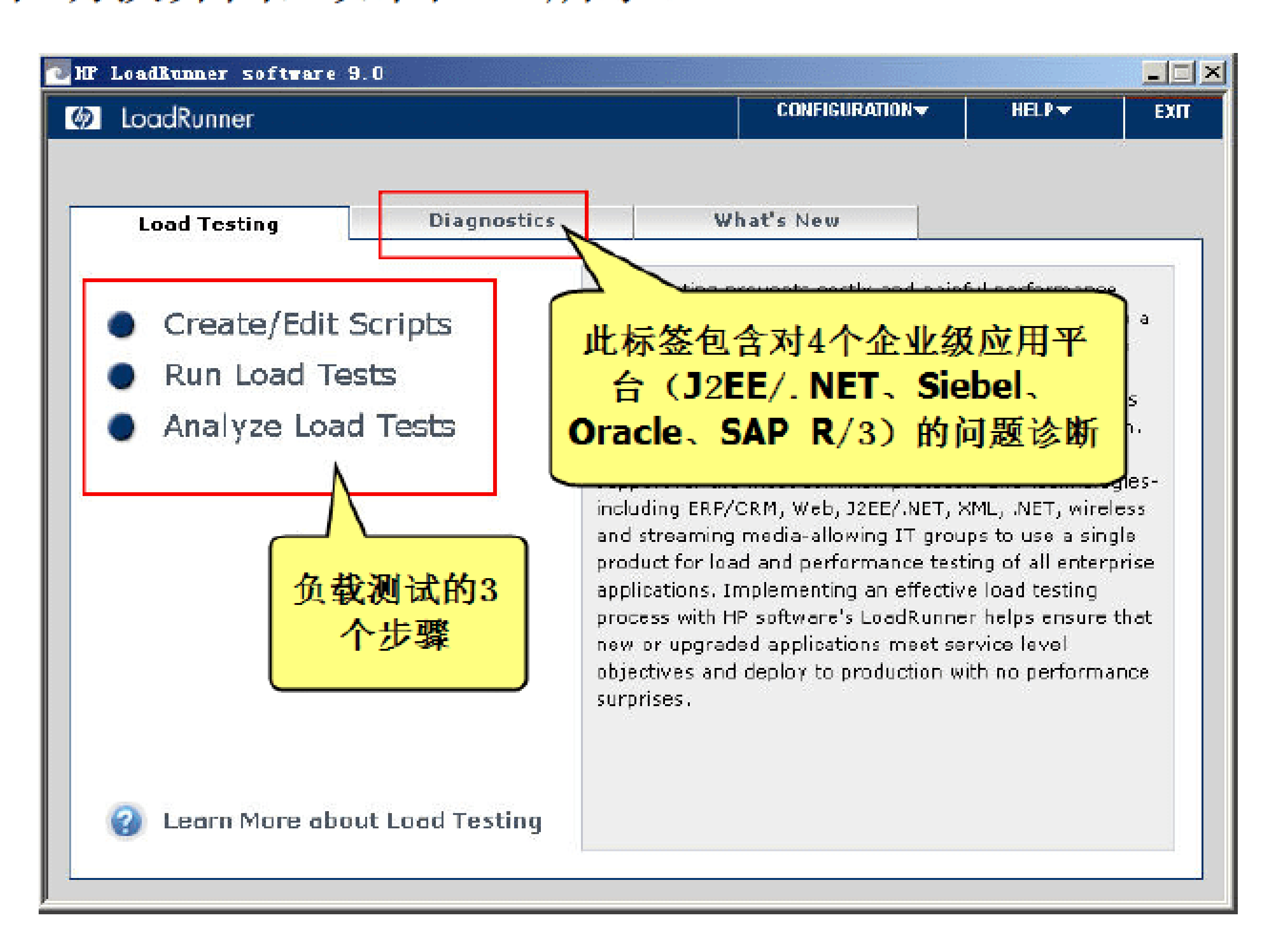


图 7-5　LoadRunner 9.0 的导航窗口

从图 7-5 中可以了解到，目前就我们所知，LoadRunner 的主要功能有如下几部分：

（1）负载测试（Load Test），其中还包含 3 个步骤：

- 创建、编辑脚本。
- 执行负载测试。
- 分析测试结果。

（2）性能诊断（Diagnostics），其中支持 4 类应用：

- J2EE/.NET；
- Siebel；
- Oracle；
- SAP R/3。

既然小白要进行性能测试，他首先感兴趣的就是负载测试标签下的内容。我们一般有这样的经验：学习使用一个软件的过程，收效最快的莫过于尽可能多地熟悉界面；尽可能

多地阅读帮助文档。小白首先单击了图 7-5 中的 Create/Edit Scripts 链接，经过小小的等待，结果打开了一个标题栏为 HP Virtual User Generator 的窗口，我们将在下面的内容中进行介绍。

7.3.2　Virtual User Generator 虚拟用户生成器

虚拟用户生成器（Virtual User Generator）窗体如图 7-6 所示。此时，小白发现在窗体的最上方是常用的 File、View 等菜单，而菜单下方有一行导航条，但都为灰色，无法使用，这是由于当前并没有打开脚本，此处的按钮均不起作用。在窗体的主体部分是开始页，其中有 3 个选项卡，分别为 Script（脚本）、Online Resources（在线资源）和 What's New（版本更新）。小白对后两个标签都不感兴趣，既然是从导航页图 7-5 中的“创建/编辑脚本”进入的当前界面，那么，直接选择 Script 标签则是很合理的行为。



图 7-6　Virtual User Generator 的开始界面

【VuGen 是什么】

细心的读者已经发现了，在运行程序后默认显示的 What's New 标签内容中，出现了 VuGen 9.0 的字样，实际上，VuGen 正是 Virtual User Generator 的简称。由于当前 LoadRunner 的版本为 9.0，因此 VuGen 的版本也就跟着称呼为 9.0 了。有了这个简称，将使得交流工作时谈论起来比较方便。因此本书和几乎所有的 LoadRunner 相关技术文档都直接将 Virtual User Generator 称为 VuGen，这一点请读者注意。

虽然小白明白了虚拟用户生成器的简称是 VuGen，但是虚拟用户是什么？它有什么作用呢？下面将回答以上这几个问题。

【虚拟用户及其作用】

好比 RPG 游戏的主人公，在 LoadRunner 的世界里，一个软件系统真实的用户被模拟成一个虚拟的用户。虚拟用户可以进行真实用户实际在被测试软件中的操作，比如，对于一个 Web 应用，虚拟用户可以进行信息注册、用户登录、页面浏览、单击链接等活动。LoadRunner 引入虚拟用户，实际就是为了创造一个模拟的真实环境，利用众多的虚拟用户，给予被测试的软件或者 Web 应用以负载，从而发现它的性能问题。

【脚本与虚拟用户的关系】

我们知道，电影剧本规定了演员在什么时候说什么话，做什么动作，并由此形成了一场又一场的戏。除非某些特别与众不同的导演，剧本一般都是先于拍摄的时候就已经写好的。LoadRunner 中的脚本与此类似，它相当于电影剧本，而我们相当于编剧和导演，LoadRunner 相当于摄影机，虚拟用户相当于演员。我们在脚本中对虚拟用户进行组织，通过它指导“演员”们对被测试软件进行操作，从而形成了一个又一个模拟应用场景，以反映真实情况下被测试软件的性能。这种脚本，也称为 VuGen 脚本。

小白在选择“脚本”选项卡后，弹出界面如图 7-7 所示。由于之前并没有进行任何脚本的编写工作，小白直接单击 New vuser script 按钮，弹出选择协议的对话框，如图 7-8 所示。注意在图 7-8 中，Web（HTTP/HTML）协议是默认选择的。

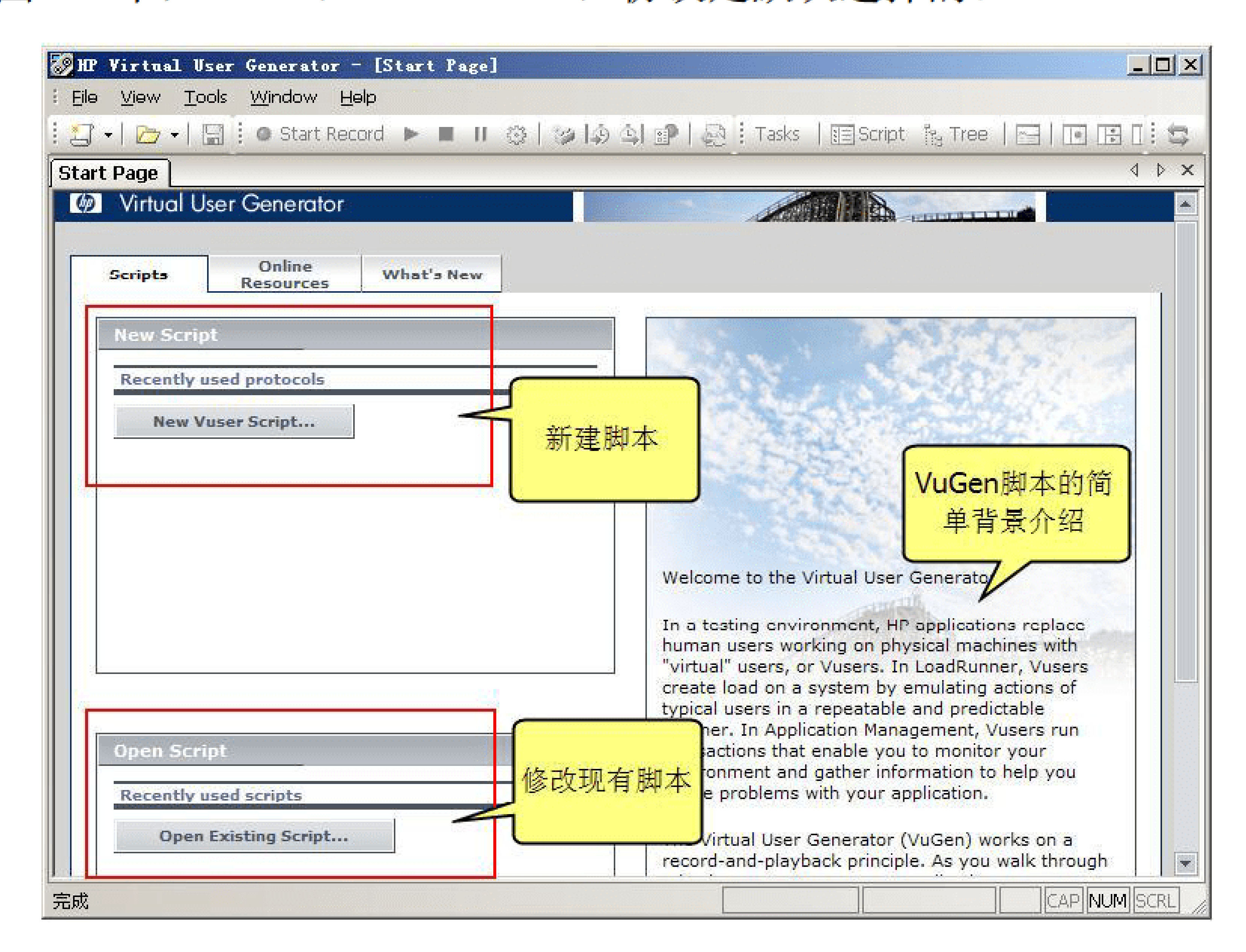


图 7-7　创建和修改 VuGen 脚本界面

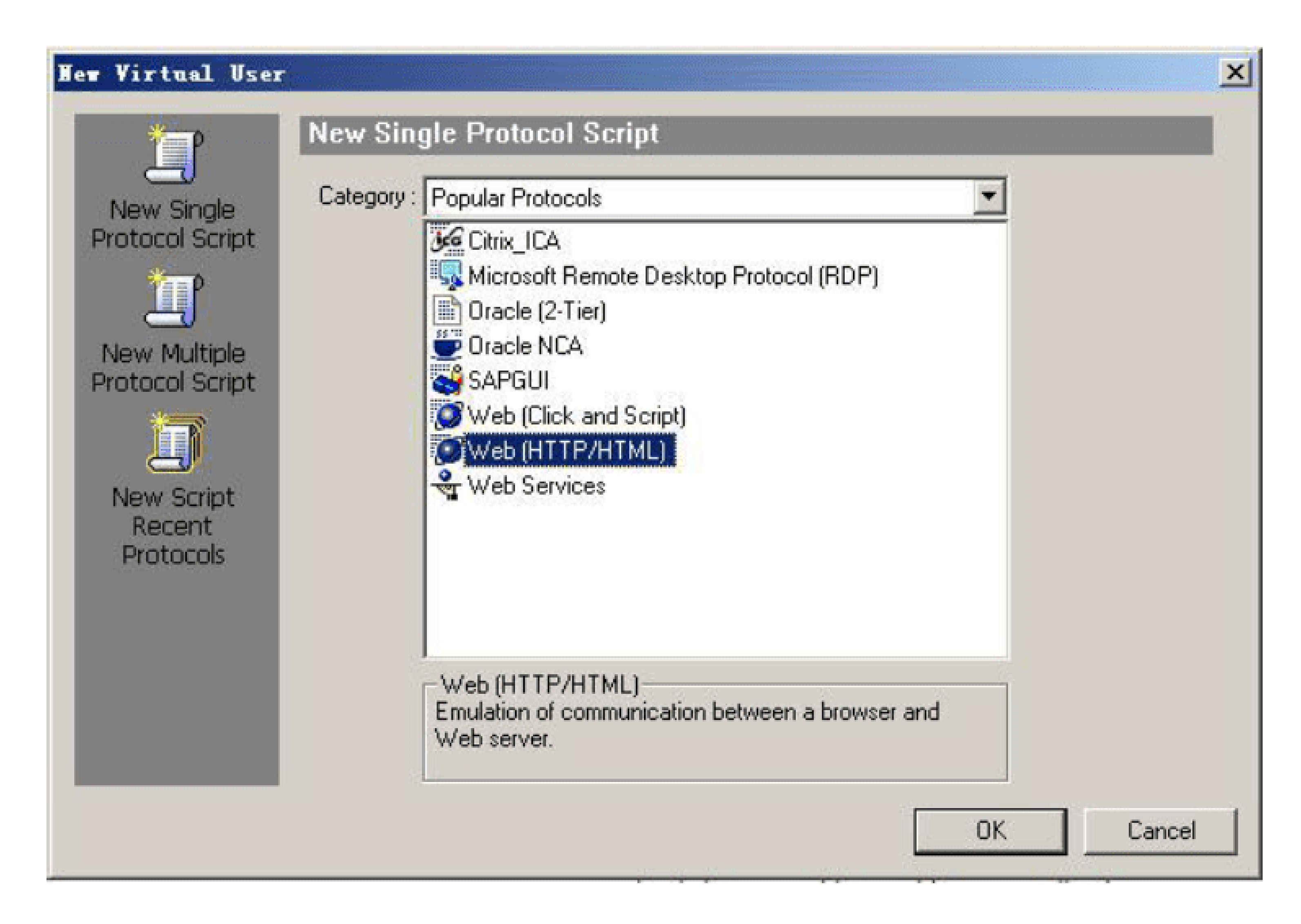


图 7-8　创建基于不同协议的虚拟用户脚本

保持该对话框中的默认设置不变，单击 OK 按钮，LoadRunner 内容页如图 7-9 所示。

图 7-9　创建 VuGen 脚本的 4 个步骤

在图 7-9 中列出了创建 VuGen 脚本的 4 个步骤，分别如下：

（1）录制步骤。在这个步骤里，首先开启录制，再以一个真实用户的身份去访问被测试的 Web 应用。停止录制后，LoadRunner 将为我们生成一系列的脚本，记录了刚才我们在 Web 应用中的操作。

（2）验证步骤。为了验证第 1 步所生成的脚本是否有效而采取的步骤。

（3）强化步骤。有的时候，单凭真实用户的操作是不够的，还需要 Web 应用后台处理的参与，比如交易处理等，这些都是界面上看不到的过程，需要加入到 VuGen 脚本中。

（4）准备负载步骤。我们在前 3 步只是得到了一个真实用户在 Web 应用上的操作，在第 4 步，我们要准备很多的虚拟用户，根据预定的性能测试目标与实施规则，对它们进行时间、数量上的调配，从而为负载测试做准备。

下面本章将对这 4 个步骤分别进行详细说明。

7.3.3　创建 VuGen 脚本 I：录制过程

所谓录制，就是对真实用户在软件或者 Web 应用中一系列操作的捕获过程。为了保证录制的准确性，如下两点必须要注意：

- 这一系列的操作必须个个清晰、容易量化。比如，登录网站、选择商品并订购、填写发货信息，提交订单这些操作都是清晰的，很好规定是否成功与否，页面响应时间也好计算。而网上购物这个操作就太笼统了。
- 这些操作作为一个场景的部分必须完整。比如还是上一点举的例子，如果从选择商品并订购开始录制，省去之前必须（是否必须视被测 Web 应用决定）的登录网站部分，那么录制成的脚本是无法得到正确结果的，因为网站尚未登录，虚拟用户的订购操作无法成功。

【VuGen 脚本的结构】

在使用默认的 Web 协议（HTTP/HTTPS）或者其他协议进行录制时，每一个 VuGen 脚本都由 3 部分组成：vuser_init、Action 和 vuser_end。Vuser_init 部分是初始化过程，对于 Web 应用来说，所包含的典型功能就是网站的登录。Action 部分代表用户实际的操作，可以由 LoadRunner 用户进一步细分为多个小部分。Vuser_end 部分是清理过程，比如退出登录、保存状态等。

1．录制脚本及其设置

了解了不少背景知识，小白有点跃跃欲试了。在图 7-10 中，他首先看到的就是界面导航条和下方两个开始录制的按钮，并迫不及待地单击了其中一个。在单击开始录制按钮后，弹出窗口界面如图 7-11 所示。它描述了录制的基本设置，我们需要在 URL Address，即网址中输入要访问的 Web 应用地址。

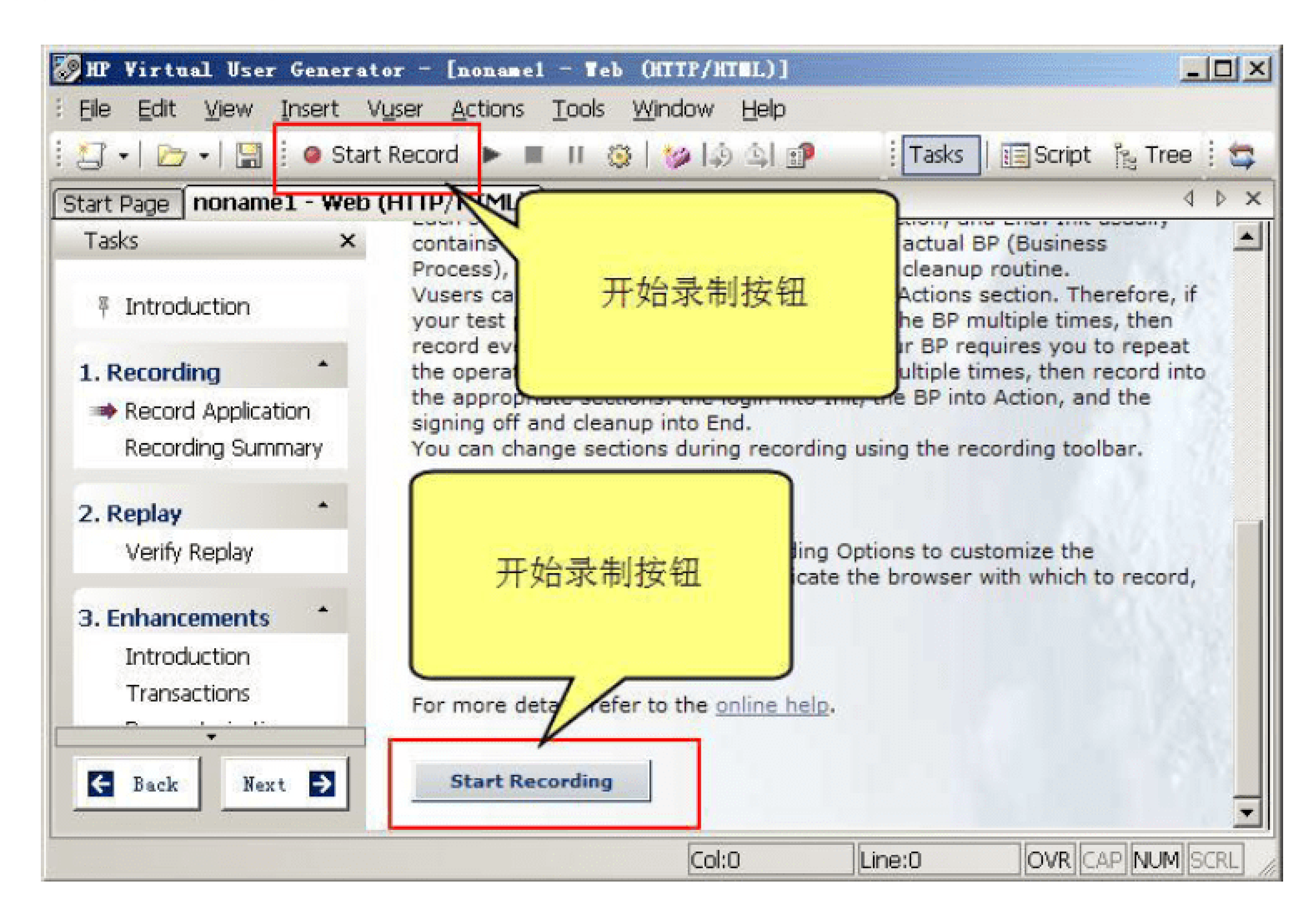


图 7-10　开始录制按钮的位置

在图 7-11 中有一个选项需要特别指出，即“录制程序启动”（Record the application startup）复选框。在默认情况下，该选项是被选中的，意味着只要被录制的程序一启动，就进行录制（在图 7-11 中被录制的程序是 IE 浏览器）。当该选项没有被选中时，被录制的程序启动并不立即开始录制，而是等待用户在需要录制的时间和位置再单击“录制”（Record）按钮，才开始录制过程。

小白决定尝试一下录制的功能，于是他在网址中输入 www.sohu.com，由于考虑到该网站可能会有一些弹出窗口、视频广告等，显示首页完成时间会比较慢，因此小白将 Record the application startup 选项清除。在一切确认填写无误后，单击 OK 按钮。

此时，一个 IE 浏览器被自动打开；同时，在桌面上出现了一个小对话框，内容显示当前录制处于暂停状态，在文字下方有 Record（录制）按钮和 Abort（中止）按钮用来改变录制状态，如图 7-12 所示。

等待浏览器打开，并且其打开在图 7-11 中输入的网址后，我们就可以选择需要的操作开始录制了。方法很简单，在一切就绪之后，单击图 7-12 中的 Record（录制）按钮即可。小白这样进行操作后，在桌面出现一个录制控制工具条，如图 7-13 所示。

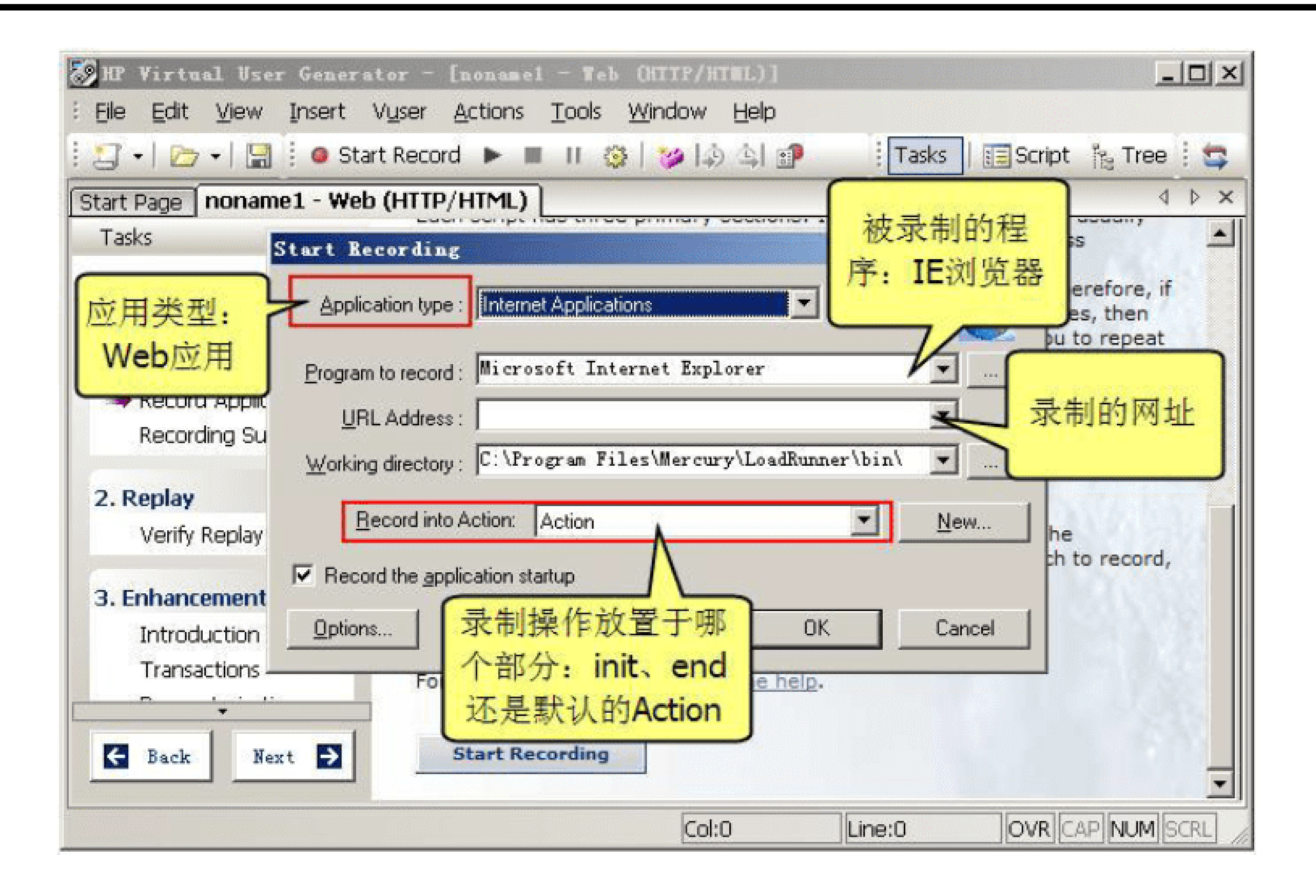


图 7-11　开始录制前的设置



图 7-12　清除启动录制选项后，录制默认处于暂停状态

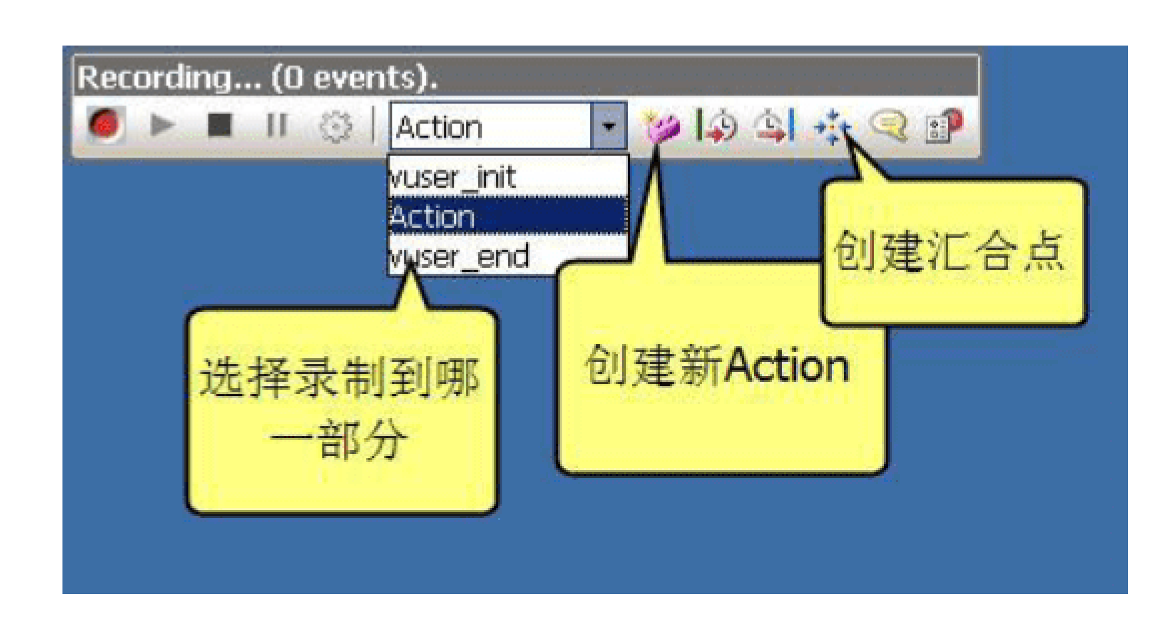


图 7-13　LoadRunner 的录制控制工具条

在图 7-13 中，我们需要注意如下几点：

- 录制控制工具条的标题栏显示了记录了多少事件（events），这里的事件指的是什么将在后面脚本生成后介绍。
- 录制按钮的设计与录音机是一样的，有录制、播放、停止、暂停按钮，使用方法无须解释。
- 录制可以选择存放的位置，如图 7-13 下拉列表框中所选择的默认位置 Action。在下拉列表框旁边，是创建新 Action 的按钮。如果单击后，将出现 Action 命名对话框，如图 7-14 所示。单击 OK 按钮后，图 7-13 中的下拉菜单将出现刚才创建的 Action 名称，如图 7-15 所示。

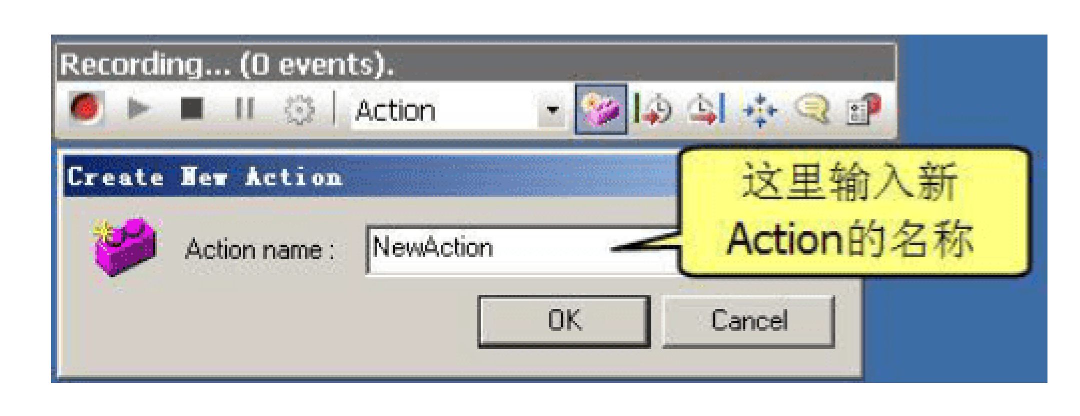


图 7-14　创建新的 Action



图 7-15　单击 OK 按钮后新的 Action 已经加入到下拉菜单中

【为什么要添加 Action】

所谓 Action，在 LoadRunner 中指的就是录制成的脚本中，虚拟用户一系列操作过程的

清单。添加新的 Action，有利于将不同类型的操作分类，方便脚本的阅读和管理。比如可以将购物的一系列操作划分为多个 Action：选购商品、收银台结算、提交订单等。

2. 脚本的编辑以及录制总结

经过一些操作，小白认为录制可以结束了，因此单击图 7-13 中的停止按钮。录制控制工具条随即消失，LoadRunner 在进行一段时间的脚本代码生成（Code Generation）之后返回脚本编辑状态，如图 7-16 所示。

在图 7-16 中可以看到，刚才新加入的动作（名为 NewAction）也在其中。对于录制的整体情况，可以通过 VuGen 左边导航菜单中的 Recording Summary（录制总结）链接进行查看，如图 7-17 所示。

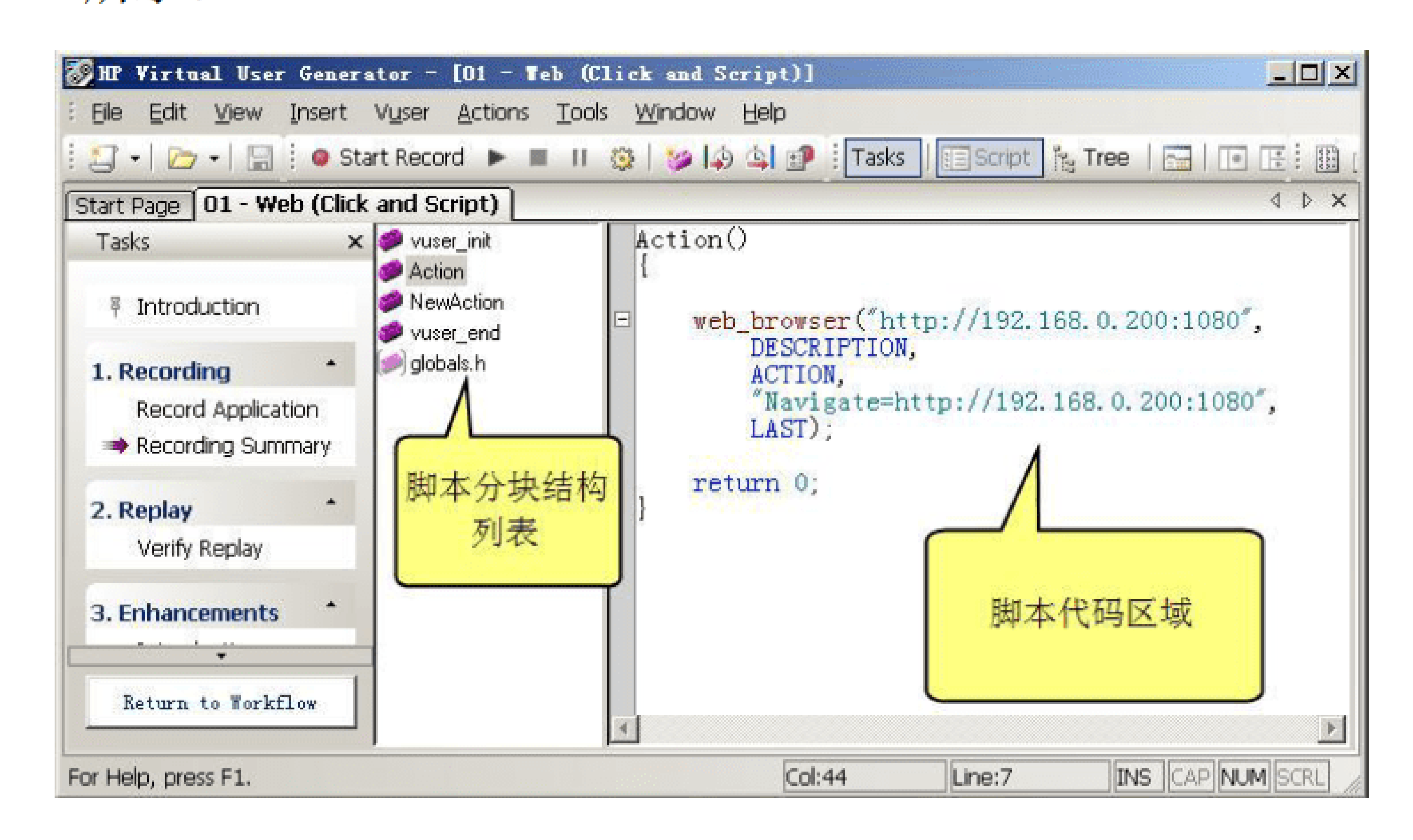


图 7-16 LoadRunner 的脚本编辑界面

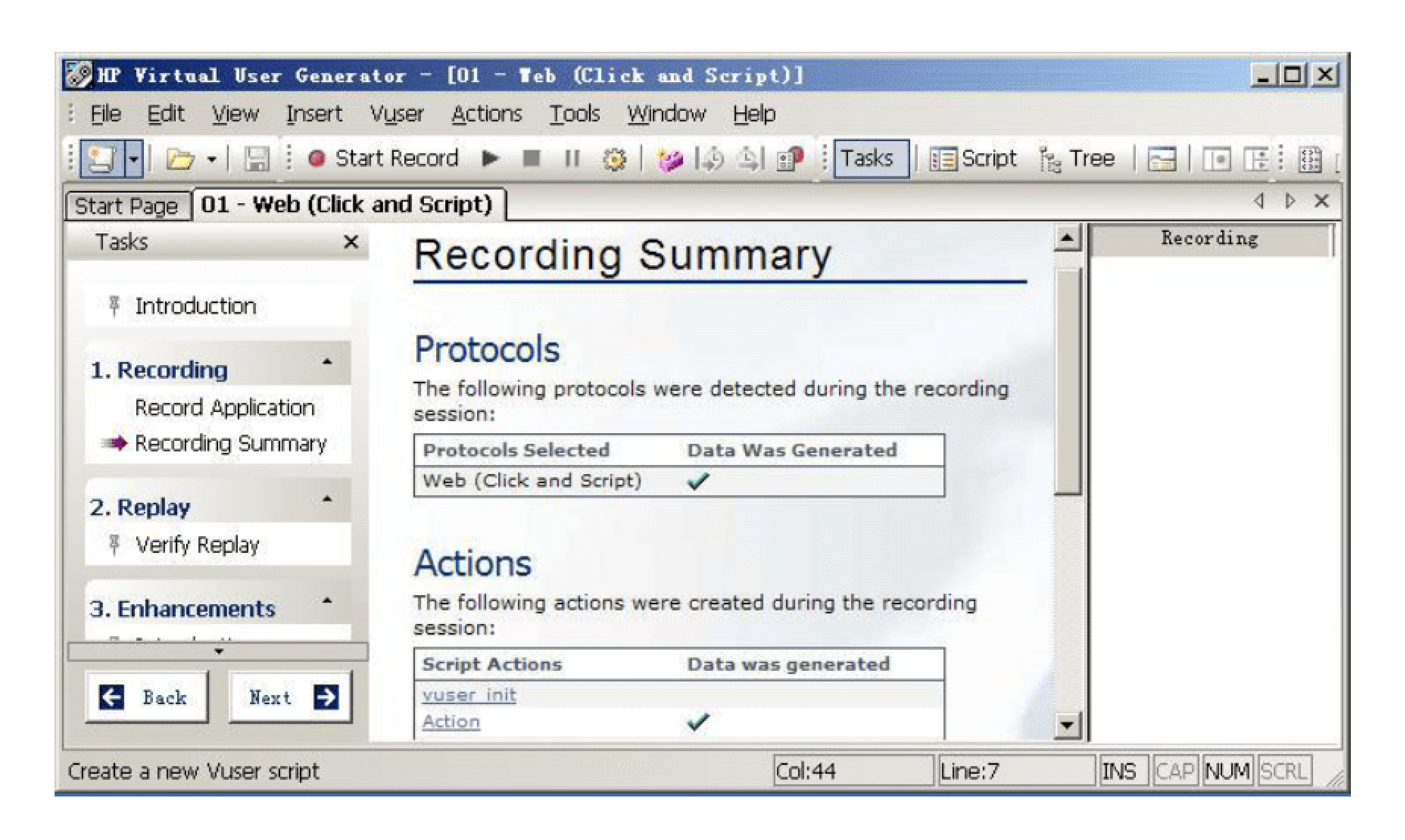


图 7-17 Recording Summary（录制总结）

我们可以在图 7-16 中的脚本代码区域对脚本进行修改，过程中可以单击界面上部导航条中的播放图标进行录制脚本的回放以确认修改的效果。这一步骤是图 7-17 左边任务栏中的回放（Replay）部分。如果认为当前的脚本已经不需要修改，则可以通过单击左边任务栏中强化脚本（Enhancements）链接或者直接单击 Next 按钮进入下一步。

7.3.4　创建 VuGen 脚本 II：强化脚本

在 7.3.3 节通过录制创建的脚本基础上，为了使得它更适合测试与结果分析的需要，有时候还可能对其进行功能强化。这就是强化脚本（Enhancements）这一阶段需要完成的任务。

对脚本的强化大致包含事务处理（Transactions）和参数化处理（Parameterization）两部分。不过，由于小白的目标在于熟悉 LoadRunner 的基本测试流程和操作，强化脚本的内容暂时先搁置在一边，等到真正进行性能测试的时候再认真仔细地进行研究。

基于以上的考虑，小白直接跳到了准备工作负荷的步骤。

7.3.5　创建 VuGen 脚本 III：准备工作负荷

前面两个步骤，不管强化脚本实施与否，都规定了虚拟用户在 Web 应用的某些行为。要知道，这只是模拟了一个用户的行为。那么如何创造出很多的虚拟用户呢？这正是准备工作负荷需要完成的工作。在 LoadRunner 界面左边的任务列表中选择 4.Prepare For Load，就可以打开这个步骤的设置，如图 7-18 所示。

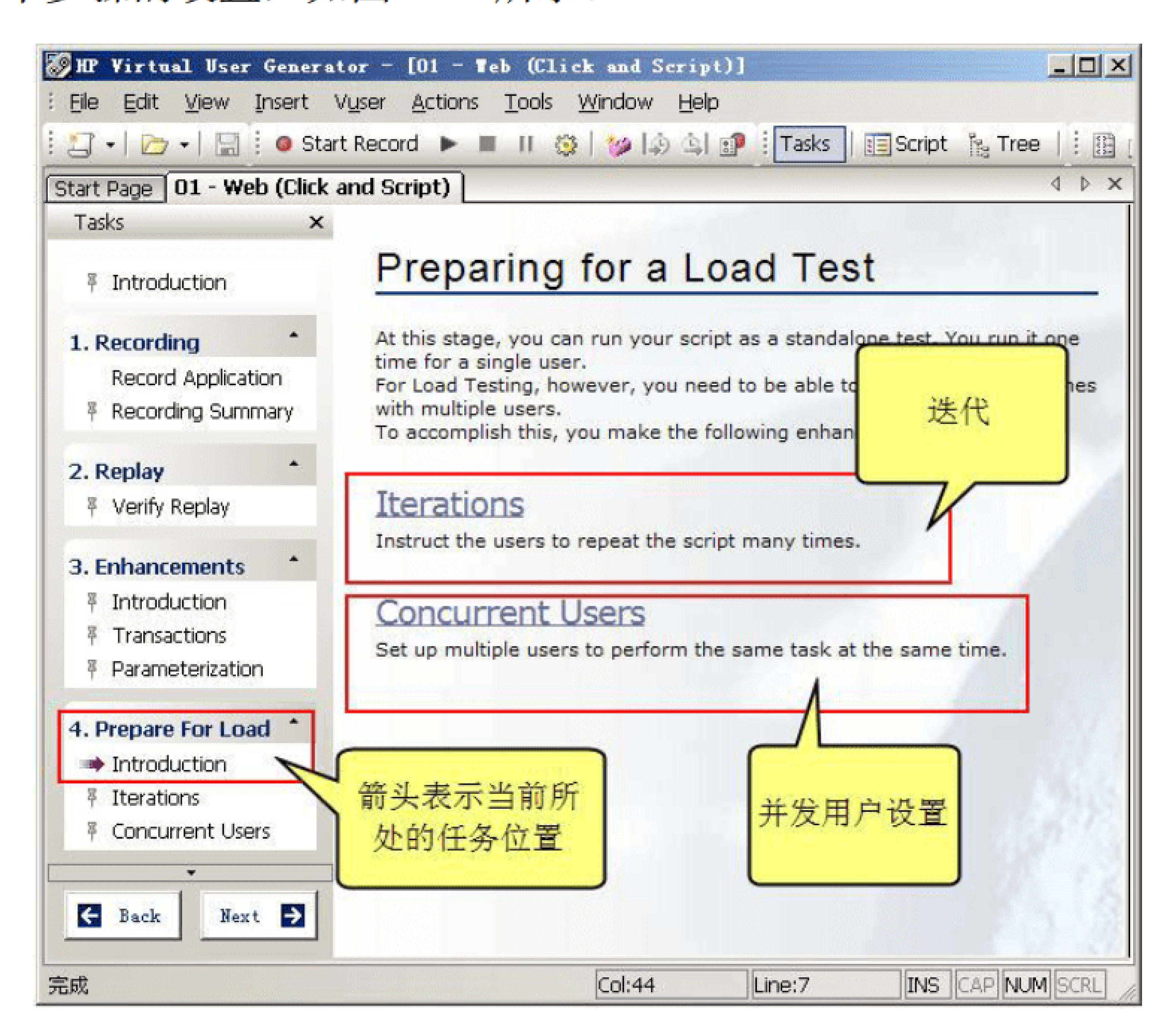


图 7-18　打开准备工作负荷任务界面

准备工作负荷阶段需要完成两类设置以实现多个虚拟用户的情况，它们分别是：

- 迭代执行（Iteration）；
- 并发用户（Concurrent Users）。

【迭代执行】

迭代执行的意思就是指某一种或者几种操作反复执行多次。比如，用户在某网站查询

从起点到终点如何乘坐公交车，每次单击查询按钮可以说是一个操作。如果用户希望查询更多其他的线路，就会多次单击查询按钮，这就是迭代执行的一个实际例子。

【并发用户】

如果在同一时刻，不同的用户同时访问 Web 应用，那么它们的全体就可以称为并发用户。并发用户一般不关心具体的用户信息，主要关注数量，因为这个数值对于 Web 应用的性能有很大影响。举个生活中实际的例子：在我国和世界各国的城市地铁入口都会有一些验票闸机，如果它们的数量较少，能同时进入地铁站台乘车的人数就较少，导致后面的乘客需要等待，地铁的服务质量下降。Web 应用也是同样的道理。并发用户在 LoadRunner 以及其他性能测试工具软件中都是一个很好的模拟真实环境进行性能测试的方法。

下面笔者将对 LoadRunner 中设置迭代执行和并发用户进行介绍。

1．迭代执行的设置

单击图 7-18 中左边导航菜单的 Iterations（迭代项）链接，右边的内容栏随即变为图 7-19 所示的外观。

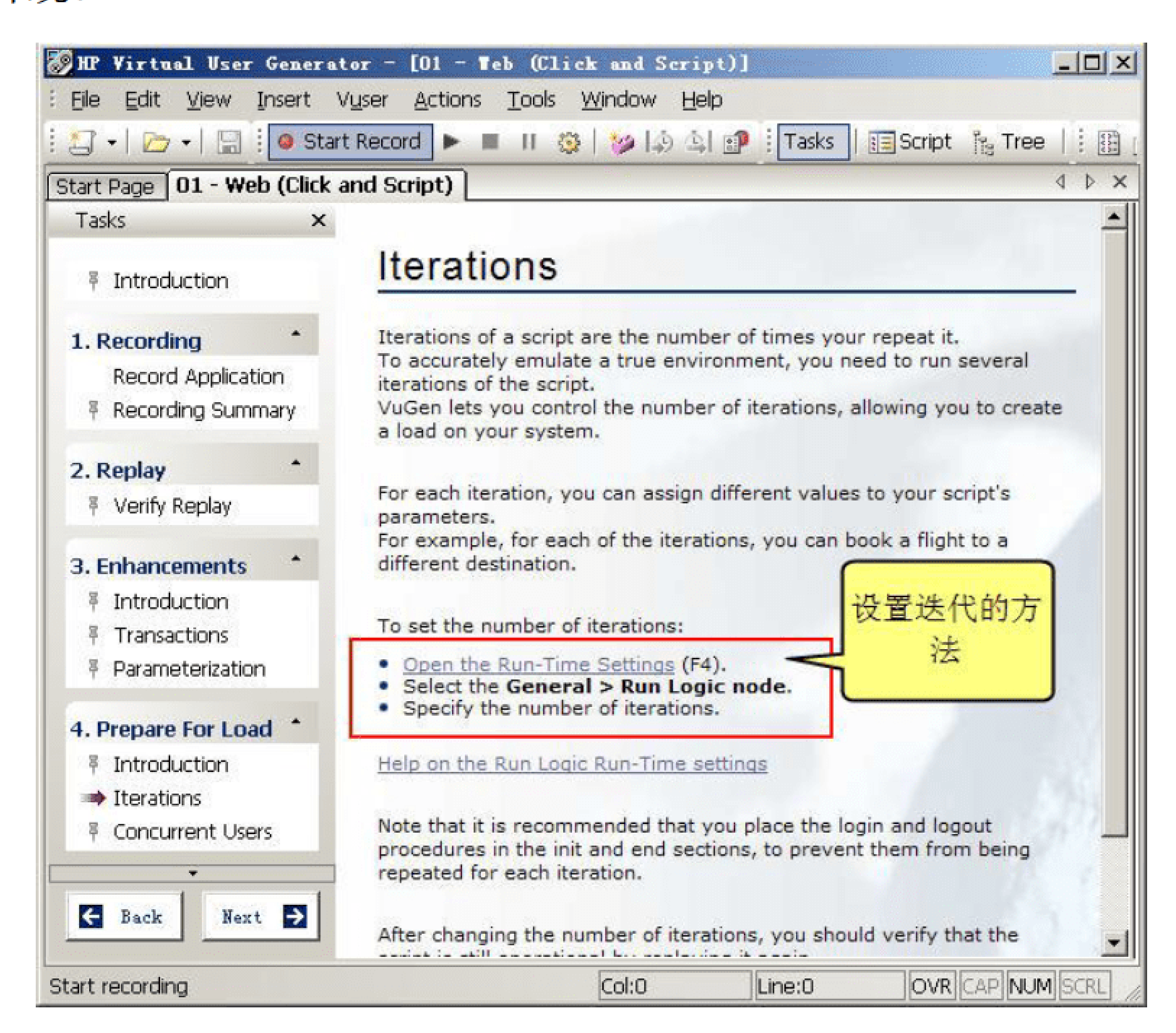


图 7-19　迭代介绍页面以及设置入口

单击图 7-19 中框住的 Open the Run-Time Settings（打开运行时设置）链接，或者直接按下 F4 键，将弹出对 LoadRunner 运行时设置的对话框。在该界面中，按照图 7-19 中的提示，选择 General（通用）节点，然后选择其下 Run Logic（运行逻辑）子项目，在右边的设置页修改 Number of Iterations（迭代次数）文本框，将其改为某个合适的数字即可，如图 7-20 所示。

2．并发用户的设置

并发用户的设置是通过对场景（Scenario）的设置来完成的。实际上，性能测试也是

基于场景来进行，不同的场景可以考验 Web 应用不同方面的性能信息。这一点，从第 5 章的性能测试分类也可以联想到。

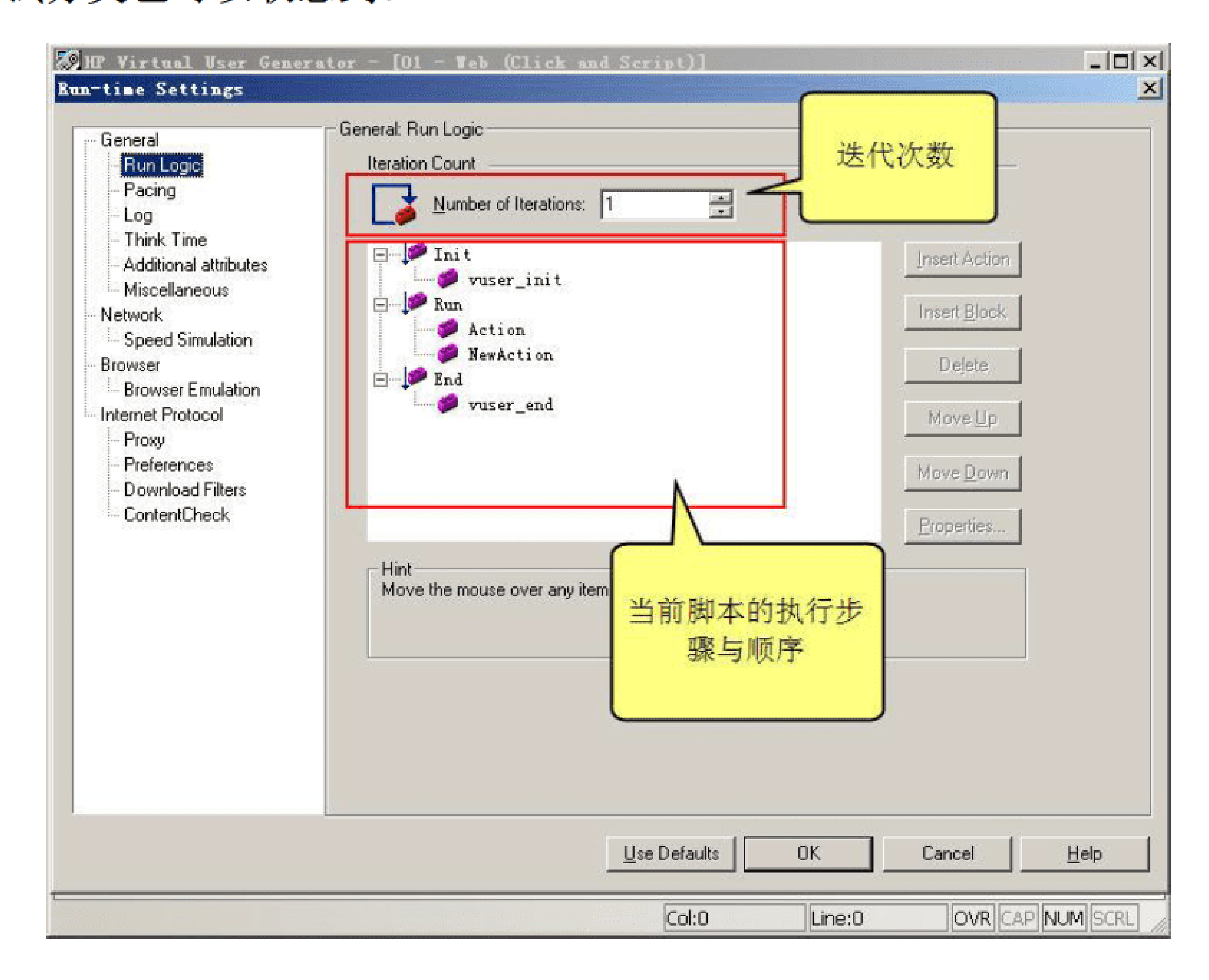


图 7-20　在 LoadRunner 运行时设置对话框中设置迭代次数

- ❑ 压力测试考察的是 Web 应用在一定压力下的性能情况，这时候的场景就是一定压力的环境。
- ❑ 持久度测试考察的是 Web 应用长期运行的性能情况，这时候的场景就是系统的长期运行。

在 LoadRunner 中，同样提供了对以上这些场景的模拟。但相对于前文所讲述的，只针对脚本运行方式的迭代操作，场景要复杂一些。每一个 LoadRunner 场景包括负载机器（Load Machine）、测试脚本、场景模式和虚拟用户运行模式。场景的创建、运行和管理通过控制器（Controller）来进行。

【控制器、虚拟用户、场景的关系】

介绍到这里，控制器、虚拟用户、场景等一系列的名词可能容易引起混淆，因此需要一个形象的解释。我们依然采用地铁入口，乘客进站这个示例来说明。在这个例子中，乘客进站就相当于一个场景，它由地铁入口（负载机器）、进站需要刷卡（测试脚本）、进站需要一定的顺序比如先来后到等（场景模式）、乘客是所有人都走一个入口还是分散进入多个入口等（虚拟用户运行模式）组成。LoadRunner 中的虚拟用户相当于地铁乘客，虚拟用户的运行模式、场景模式都需要由控制器来控制，它相当于地铁站台的管理和调度人员。

单击图 7-21 中所示的 Create Controller Scenario（创建控制器场景）链接，即可进入控制器设置窗体（已经显示在图 7-21 中）。

在图 7-21 中出现了场景的类型，有如下两种。

- ❑ 默认选择的 Manual Scenario（手工场景）单选框：这个场景需要我们完全手动地设置，指定虚拟用户数量、指定每个虚拟用户的运行时间、产生负载的机器等，

见图 7-21 中创建场景窗体下方的几个文本框。

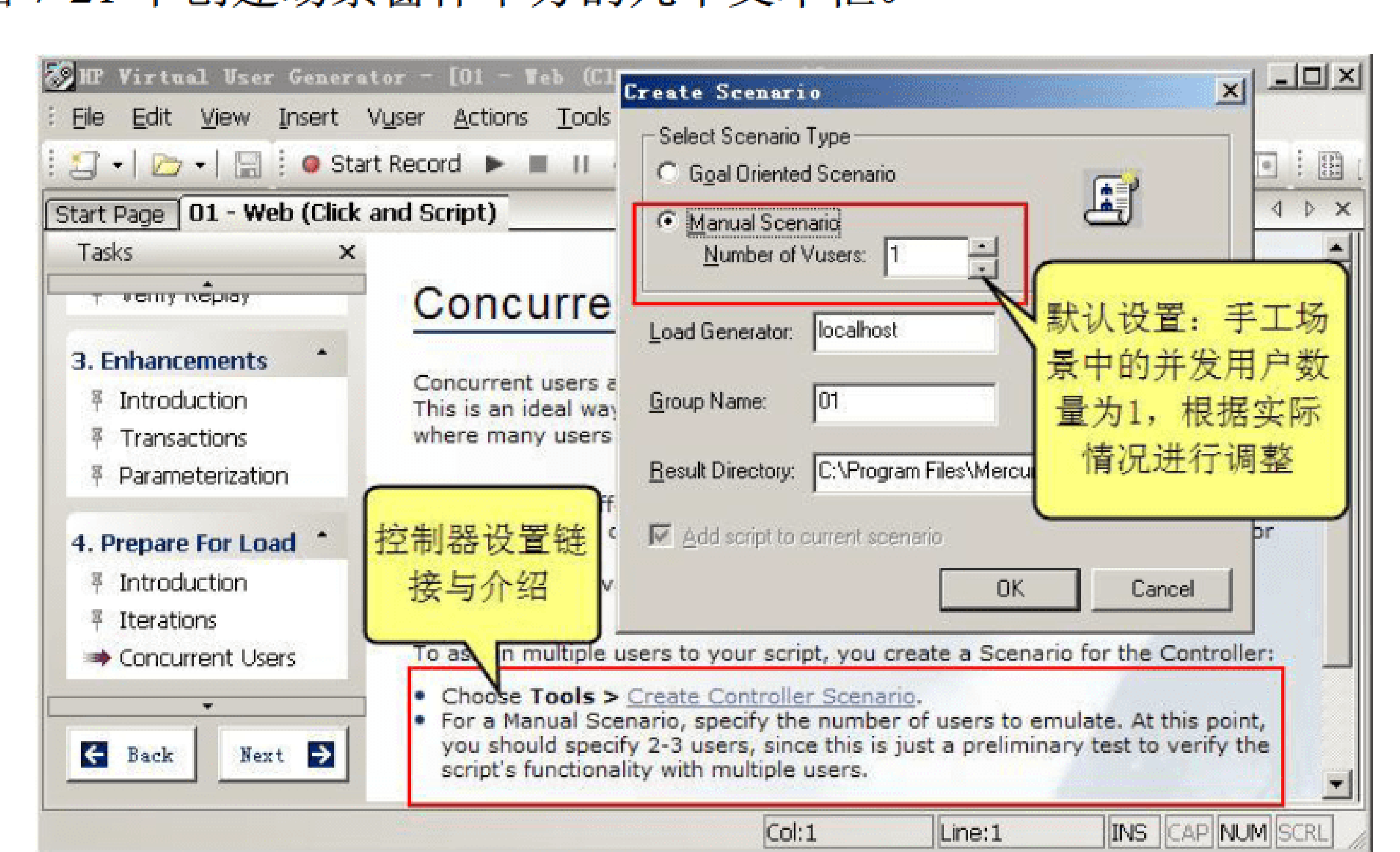


图 7-21　创建场景窗体的打开方法与并发用户设置

- Goal Oriented Scenario（目标场景）单选框：这个场景需要我们设置性能测试要达到的目标以及虚拟用户的增长模式。当 LoadRunner 执行场景时，如果达到了此处设置的目标，就会停止。

由于本章的目的只是为了帮助小白尽快熟悉 LoadRunner 的基本菜单、概念和操作流程，场景的进一步讲述将在第 10 章进行。在这里，我们只需要了解并发用户数量的设置可以通过图 7-21 中框出的地方进行即可，并不需要详细进入场景的设计阶段，因此单击 Cancel（取消）按钮回到主窗体。

7.3.6　创建 VuGen 脚本 IV：完成阶段

到这里为止，本章已经把创建 VuGen 脚本的基本步骤介绍完毕，相当于到达了 LoadRunner 左边任务栏中的完成（Finish）阶段。下面笔者来总结一下创建 VuGen 脚本的目的和步骤：

【创建 VuGen 脚本的目的】

创建 VuGen 脚本的目的，就是为了创建虚拟用户以及它在被测试 Web 应用中的操作模式。进行性能测试，必须关心以下两方面的问题：

- 单一用户在 Web 应用上进行什么样的操作。
- 有多少用户在 Web 应用上同时进行这样的操作。

创建 VuGen 脚本，就是为了达到模拟以上两点的目的。对于单一用户的操作，可以通过录制脚本创建者对于 Web 应用的操作来实现。对于多个用户，可以通过创建场景来实现。

【创建 VuGen 脚本的步骤】

综合前文，创建一个 VuGen 脚本的步骤如下：

（1）用户操作的录制阶段。

（2）录制所得脚本的验证阶段。

（3）录制脚本的增强阶段。

（4）工作负荷的生成阶段。

值得注意的是，单单创建操作脚本（即截止上述第 3 步的结果）是不够的，一定要设置场景来模拟多个用户的行为，这样才能构成真正意义上的性能测试。

【VuGen 脚本如何验证正确性】

实际上，任何的步骤，都可以通过 LoadRunner 提供的 Play（播放）功能来对脚本进行验证。验证的结果，LoadRunner 会以报告的形式显示出来，如图 7-22 所示。

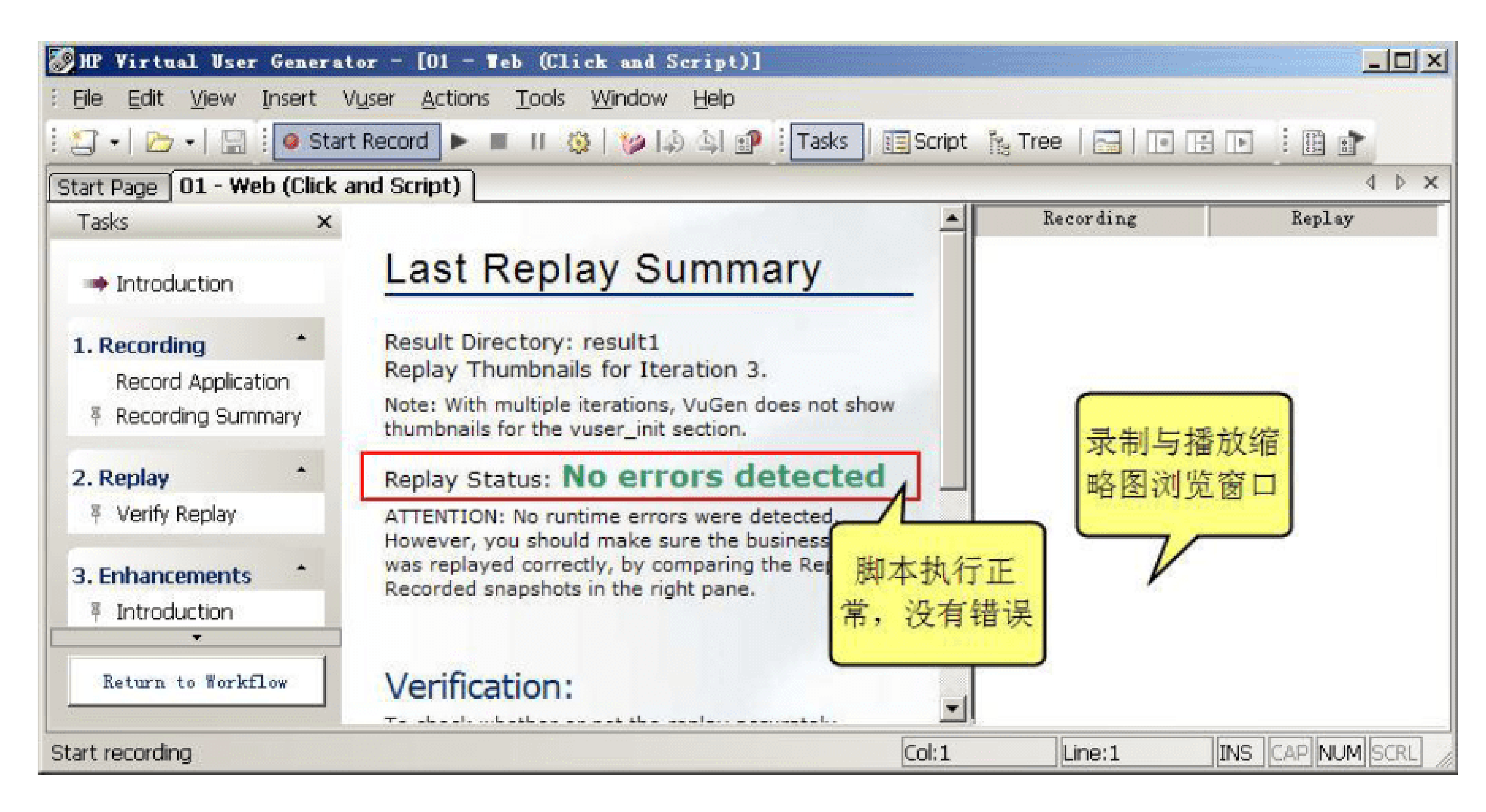


图 7-22　脚本播放结果（Replay Summary）页面

下面小白将对 LoadRunner 自带的一个示例站点进行脚本的录制和播放，进一步熟悉以上所述的基本操作过程。

7.3.7　创建 VuGen 脚本 V：利用示例站点录制一个脚本

如果读者仔细观察，就会发现 LoadRunner 在安装之后附带了很多的实用工具，它们可以通过依次单击 Windows 的“开始”|“程序”|LoadRunner 命令，在其中一一找到，如图 7-23 所示。

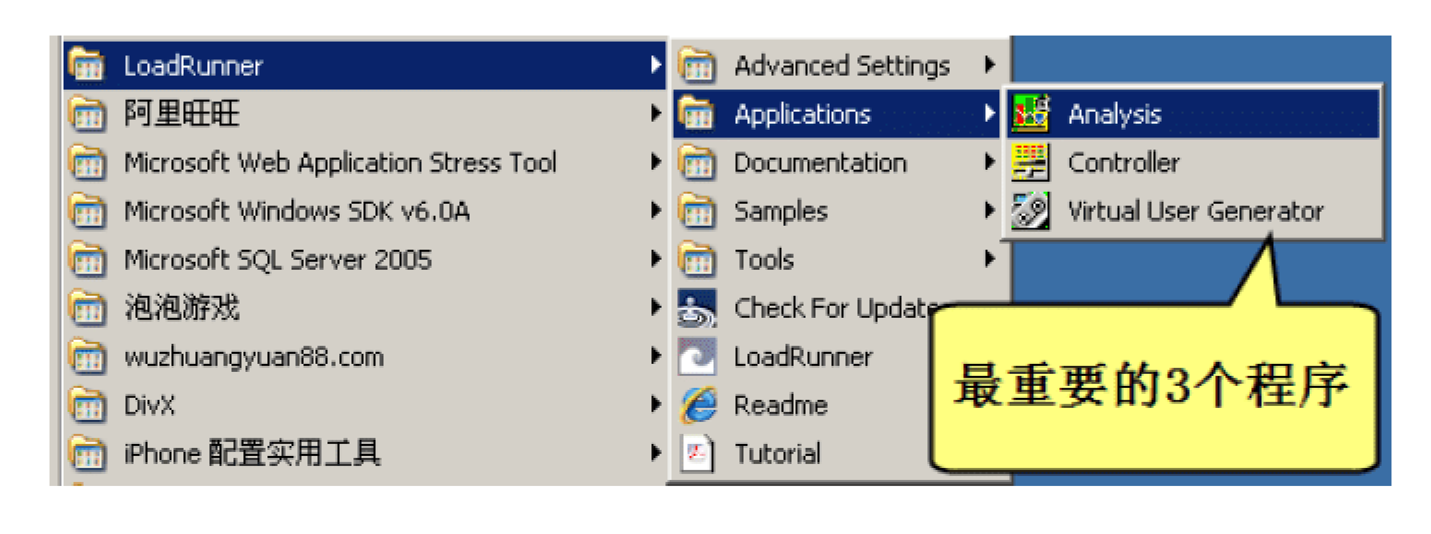


图 7-23　LoadRunner 程序组包含的内容

（1）在 Applications（应用程序）程序组中包含的 3 个程序是我们今后工作中使用最多的，分别为本章前文一直介绍的 VuGen 虚拟用户生成程序、第 10 章将要详细介绍的 Controller 场景生成和管理工具，以及后面章节要详细介绍的 Analysis 测试结果分析工具。其他程序组分别包含了若干实用工具，虽然都是辅助前文提到的这 3 个程序之用，但完全可以单独运行。

（2）在图 7-23 中，我们还注意到有一个 Samples（示例）程序组，其中 Web（网络）子菜单中包含了两个菜单项，分别如下。

- HP Tours Web Application：这是惠普公司为了方便初学者学习所提供的一个在线订票示例站点，也是本节中小白即将对其做性能测试实验的网站。
- Start Web Server：启动上述站点的命令行快捷方式。

（3）单击第二个菜单项 Start Web Server，可以发现桌面会短暂的弹出一个命令行窗口，然后在 Windows 右下角的系统任务栏中出现了一个类似 X 字母的图标，这表明 LoadRunner 所自带的网站服务器已经启动。该服务器为示例订票网站（HP Tours Web Applications）提供了支持，以供使用者学习和测试。我们首先通过单击 Samples（示例）程序组中的 HP Tours Web Application 菜单项，此时 LoadRunner 将自动打开 IE 浏览器，并调入该网站以供浏览。经过很短暂的打开浏览器的过程，示例网站显示了出来，其界面如图 7-24 所示。



图 7-24　范例站点的首页

（4）从图 7-24 中的浏览器地址栏中可以发现，该 Web 应用运行在本机 127.0.0.1、端口 1080 上。在初次访问时，由于尚未创建用户，无法登录系统，因此先单击 Sign up now（用户注册）链接至少创建一个新的用户。这里假设用户名为 xiaobai，密码也是 xiaobai。利用这个用户登录系统，单击左边的 Flight（航班）按钮，在右侧的框架网页中出现航班订票界面，如图 7-25 所示。

（5）选择和输入正确的信息之后，单击 Continue（继续）按钮，就会出现具体的航班列表，如图 7-26 所示。任意选择一个，单击 Continue（继续）按钮，出现订单生成界面，如图 7-27 所示。

（6）在图 7-27 中单击 Continue（继续）按钮，系统将列出已订航班机票的详细信息和花费，如图 7-28 所示。

至此，小白完成了一次订票的操作，如果需要订下一张票，只需要重复刚才的若干步骤即可。

【如何对该订票网站进行性能测试】

如果考察该订票网站的性能，则至少要考虑多个并发用户在类似小白这样的订票操作

下，Web 应用的响应时间等性能计数器的变化趋势会如何。因此，就需要通过录制创建描述用户订票操作的脚本，通过控制器创建场景，执行性能测试，最终分析结果。

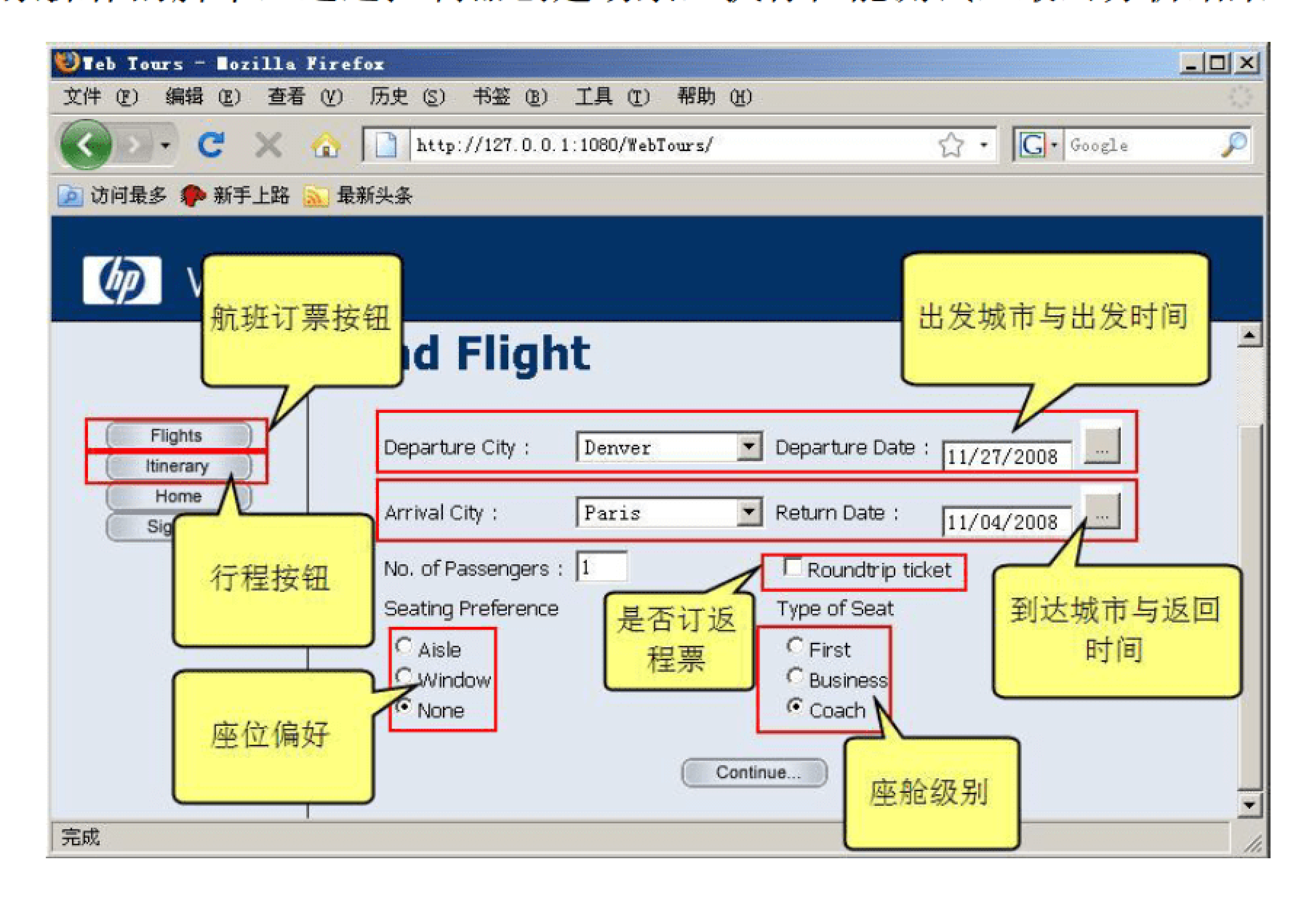


图 7-25　航班订票的主要界面

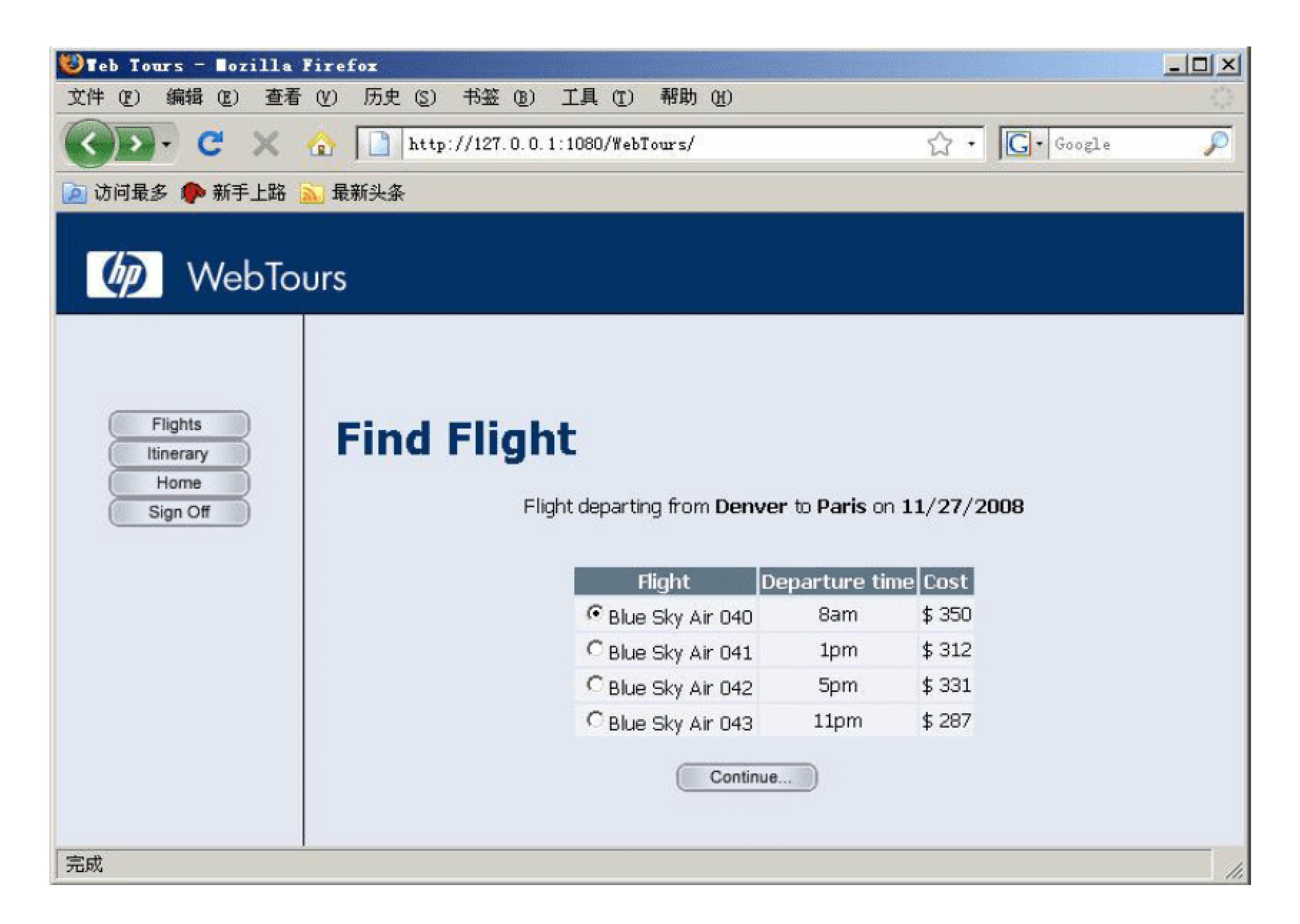


图 7-26　选择航班界面

下面小白将进行一次录制，将前文所述的操作转化为 VuGen 脚本。在这个过程中，由于前文已经讲述了基本情况，就不再重复。不过有几个要点需要注意。

【LoadRunner 中 Web 的两个协议】

当创建新的脚本时，LoadRunner 会提示选择使用一种协议进行录制。我们可以看到在图 7-8 中，列出了 2 个包含 Web 文字的协议：默认选择的 Web（HTTP/HTML）和 Web（Click and Script）。那么这两个协议有什么区别呢？

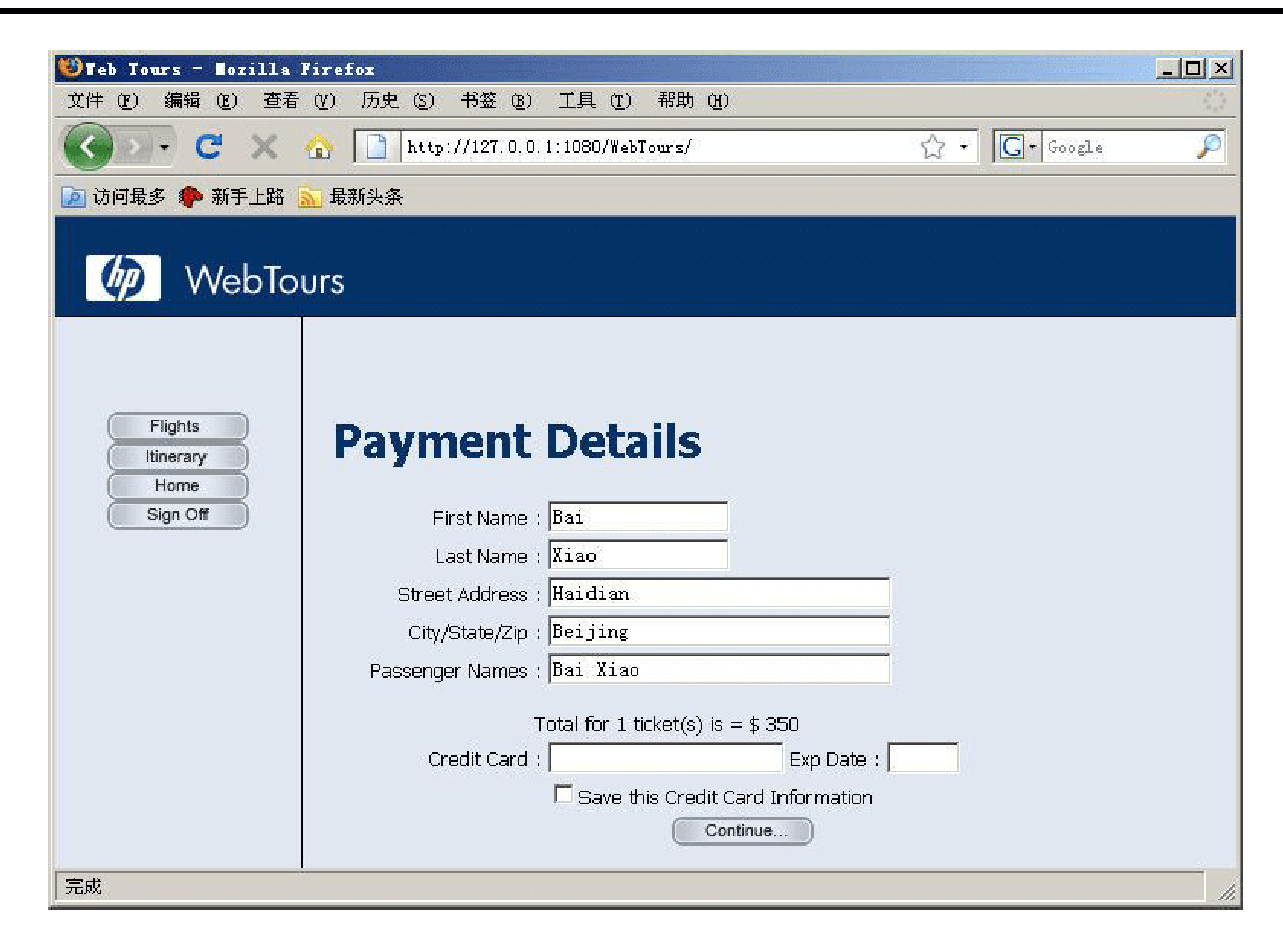


图 7-27　订单生成界面

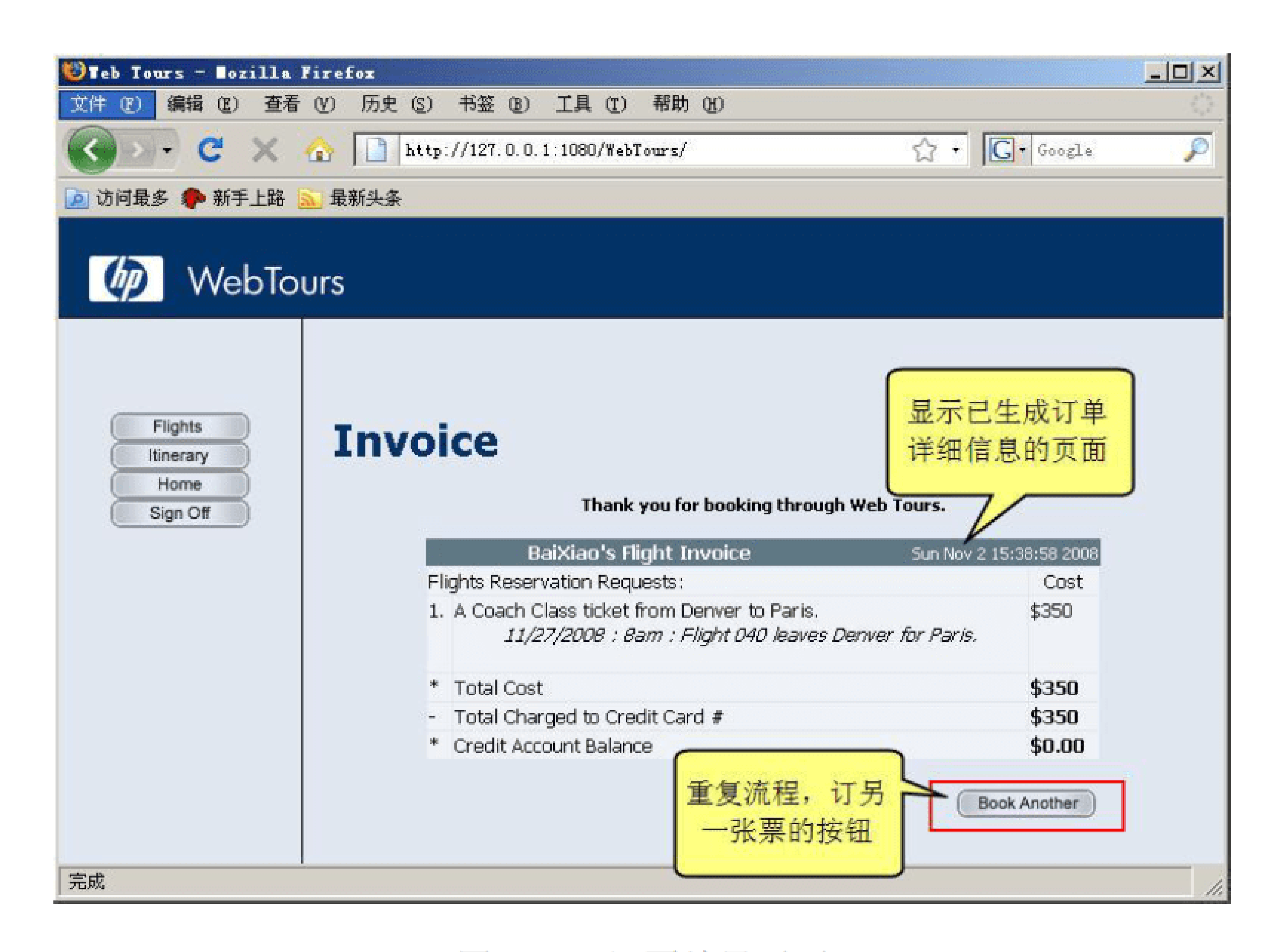


图 7-28　订票结果页面

简单地说，Web（Click and Script）生成基于图形界面（所谓的 GUI）的、用户实际操作界面过程的脚本，面向操作过程；而 Web（HTTP/HTML）生成基于浏览器向服务器发出的 HTTP 请求以及服务器的响应的脚本，面向数据。感兴趣的读者可以利用惠普提供的范例页面进行两种协议的录制以查看具体的差别。我们在后面的内容中还会提到这两个协议在代码上的区别。

【对不同浏览器实现录制】

图 7-11 中，用户操作被录制的程序是默认的 IE 浏览器，如果需要测试其他浏览器，我们可以通过单击图中文本框旁边的浏览框，选中其他浏览器的可执行程序（exe）文件即可。具体操作如下：假设小白要录制 Firefox 浏览器下的用户操作，只需要如图 7-29 那样

找到 Firefox.exe 的位置，然后设置其他选项（注意 URL 地址为：http://127.0.0.1:1080/WebTours），单击 OK 按钮即可。

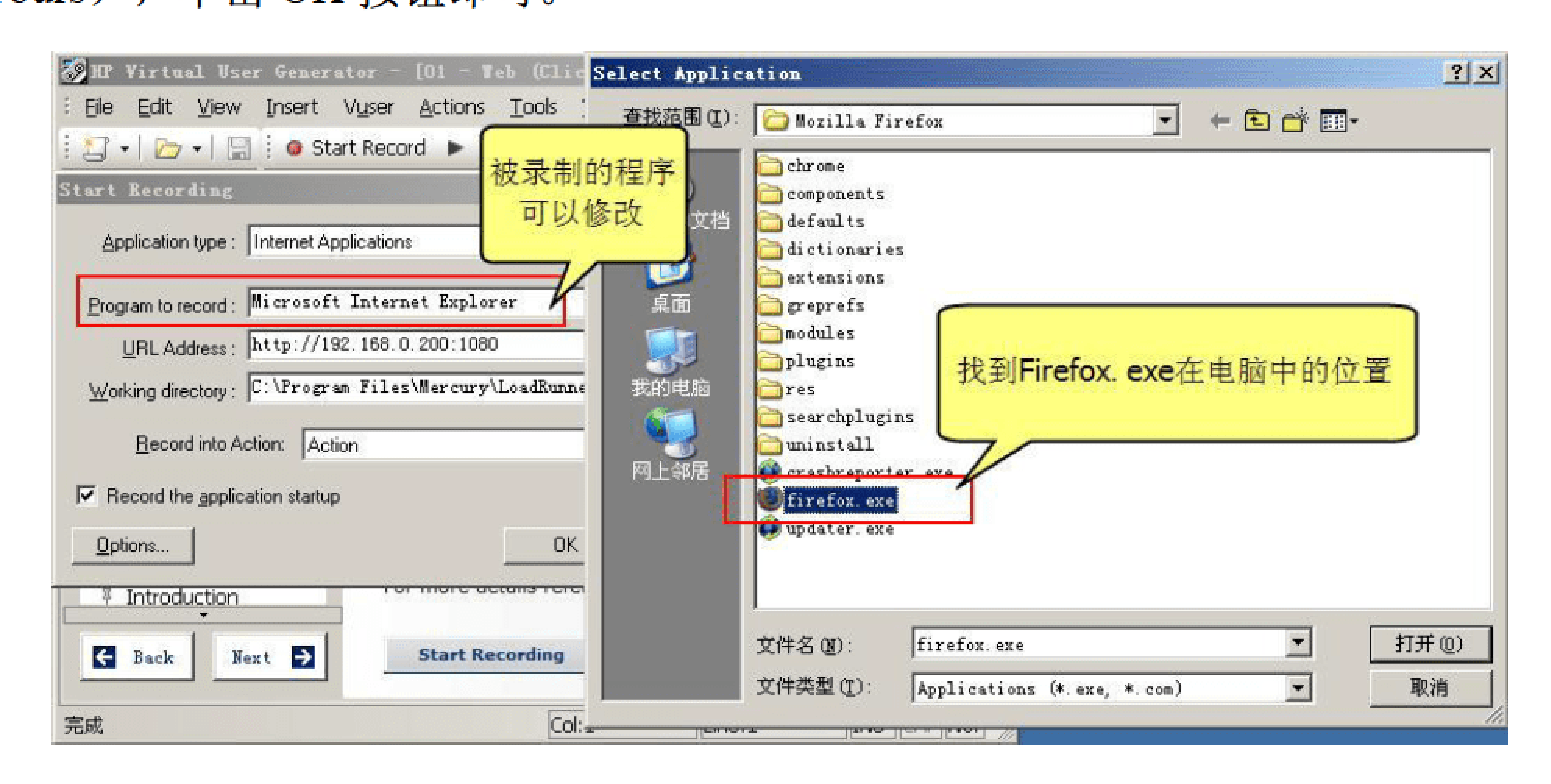


图 7-29　更改被录制的程序

（1）在录制正式开始后，小白随即开始进行一遍前文所述的订票操作，直到类似图 7-28 的界面出现为止，则停止录制。经过较短的时间，LoadRunner 自动生成了录制后的脚本，并显示录制总结界面，如图 7-30 所示。

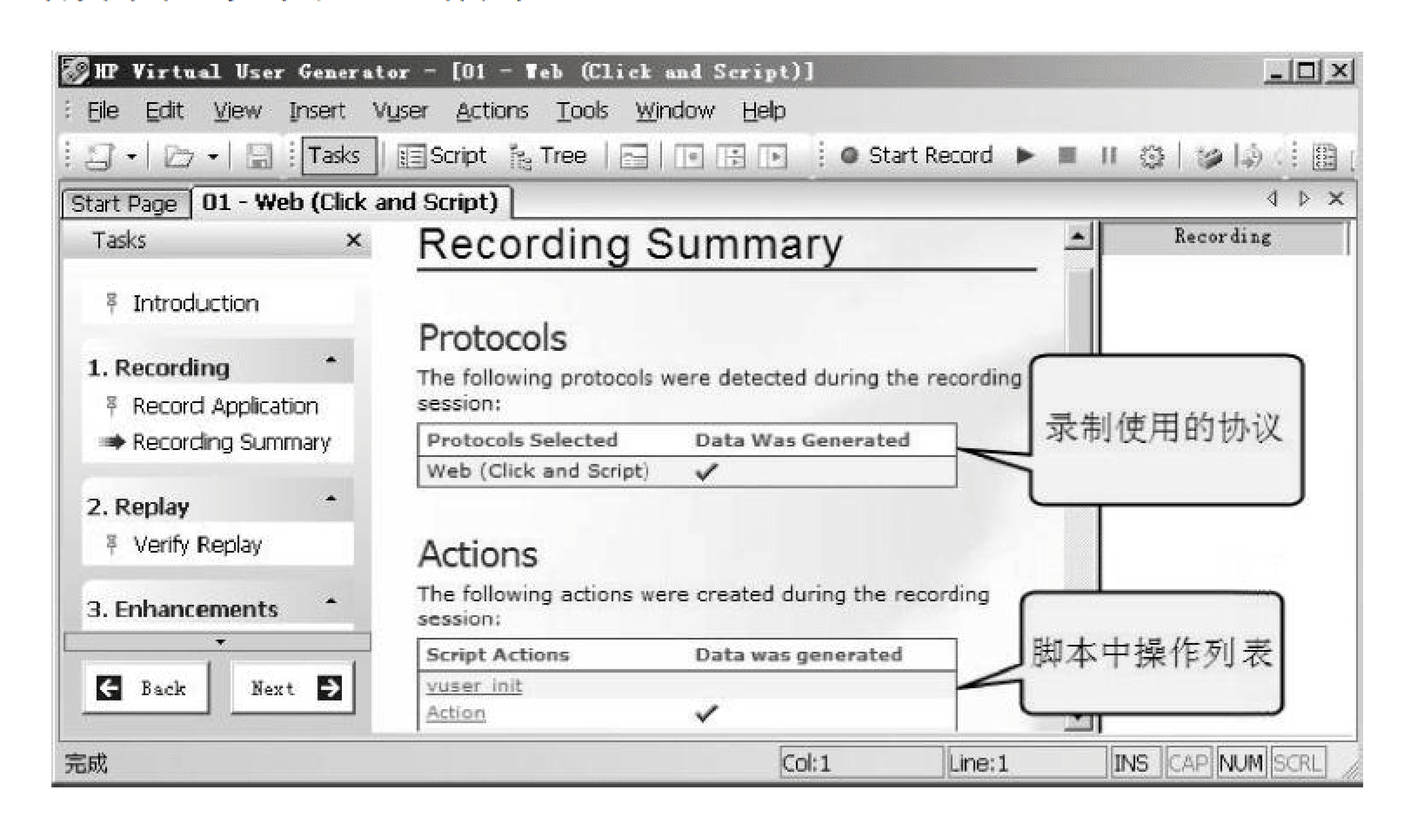


图 7-30　录制总结页面

俗话说："一口吃不了一个胖子"，LoadRunner 是一个比较复杂的软件，需要多使用多体会。由于本章小白的实验目的也只需生成一个可用的脚本即可，后面的强化脚本、迭代、并发用户以及场景的生成部分，我们留待之后的章节中进行学习。

（2）该脚本在代码编辑界面显示如图 7-31 所示。注意，这是使用 Web（HTTP/HTML）协议进行录制的，所以代码中记录了很多浏览器与服务器交互的数据。

在图 7-31 所示的代码中，我们发现在各个主要的操作之间，都有 lr_think_time 这样一个函数，它表示的是 LoadRunner（在函数名称中简写为 lr 开头）中 Think Time 时间的设置，括号内的数字单位是秒。

【思考时间 Think Time】

所谓 Think Time，中文直译成思考时间，是模拟真实用户在实际浏览网页的操作过程

中空闲的等待时间。举实际的例子来解释，我们在浏览网页或者对 Web 应用进行操作的时候，不可能是连续的，单击了一个链接之后一秒不停立刻开始浏览，浏览完毕之后立刻进入下一页等操作。每个用户在操作与操作之间总会有一定的间隔，在这之间所消耗的时间就是 Think Time，思考时间。在思考时间内，用户并没有操作，因此对 Web 应用的服务器性能没有增加影响。

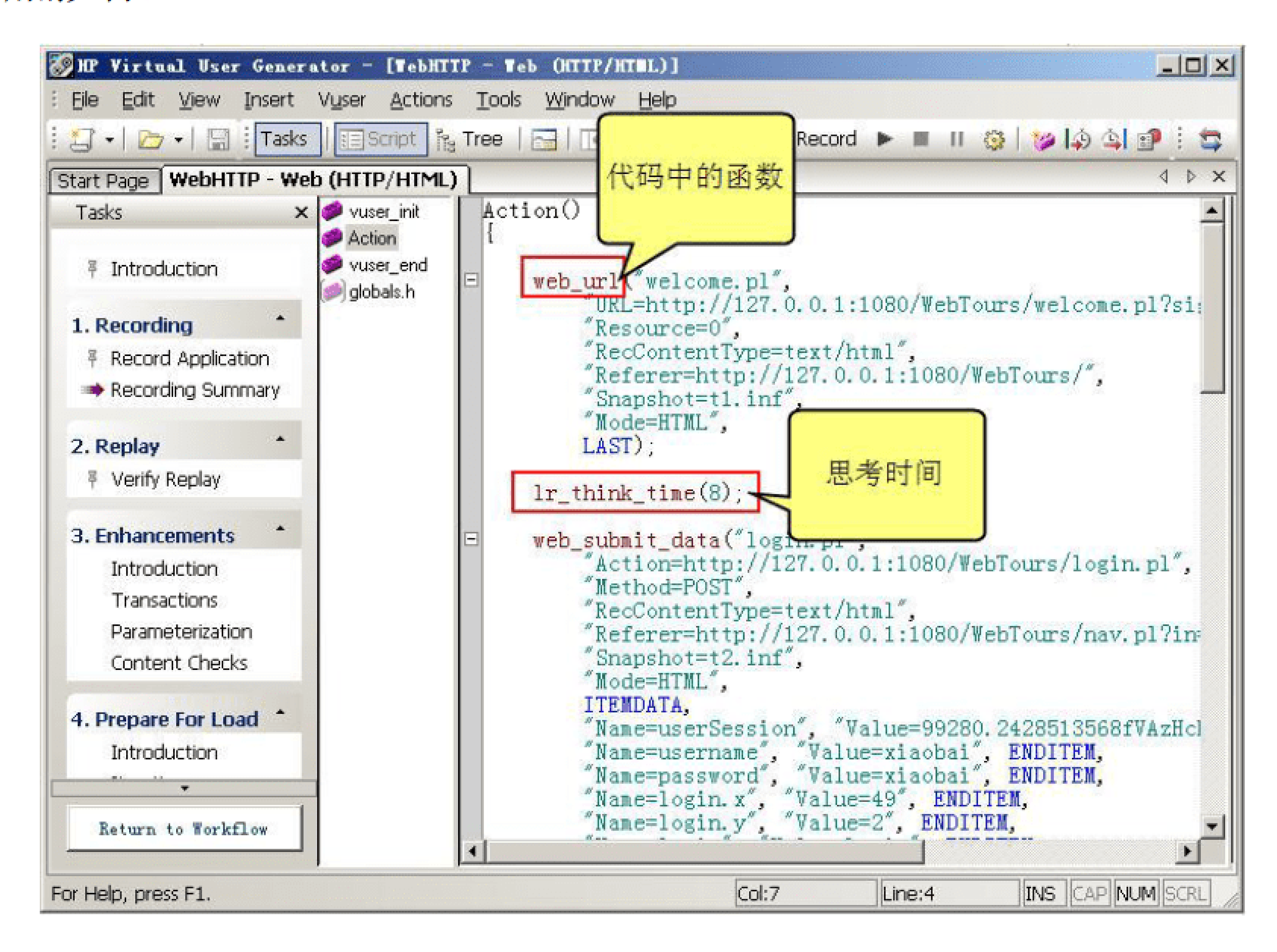


图 7-31　应用 Web（HTTP/HTML）协议录制的脚本

到目前为止，脚本录制成功，下面需要经过验证脚本的步骤，以确保当前的脚本可以播放成功。单击播放按钮，LoadRunner 却提示不成功，同时在界面下方出现了包含红色文字、表示错误信息的日志窗口，如图 7-32 所示。

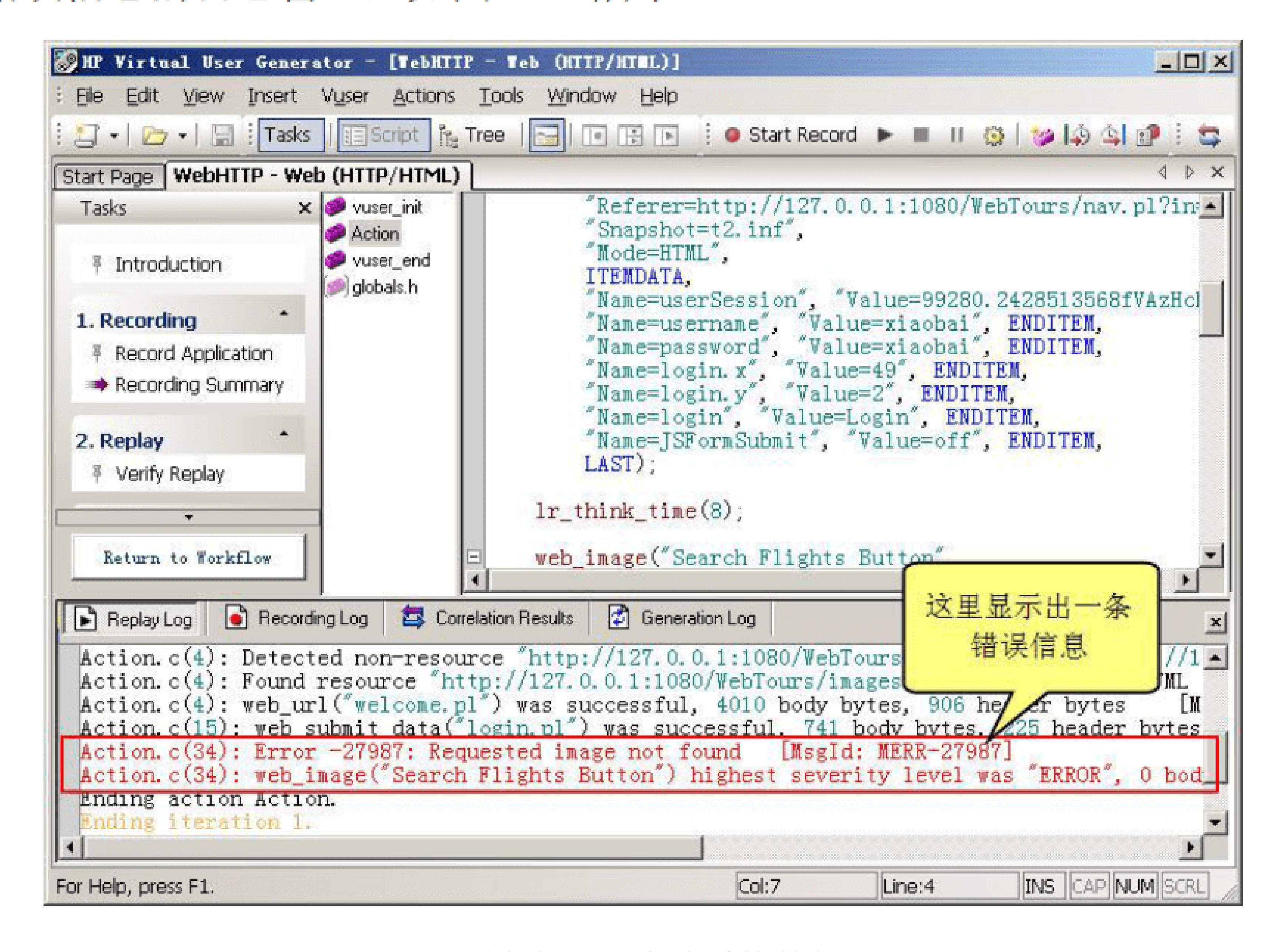


图 7-32　播放不成功时系统的提示界面

经过分析，小白发现，错误出现在 Action 部分的 34 行（即图 7-32 所示代码中的最后一行），即找不到提供航班订票的 Flights 按钮图片。那么为什么录制的时候可以成功，而播放的时候就无法成功了呢？这就需要了解 Web 应用中关于 Session 的知识，并用 LoadRunner 中的关联（Correlation）加以解决。我们将在 7.3.8 节进行讲解。

7.3.8　创建 VuGen 脚本 VI：录制脚本失败原因分析与会话

7.3.8 节小白录制了一个登录后订票的脚本，但为什么播放的时候会失败呢？在本节我们将帮助小白分析原因，找到解决办法。

如果我们仔细查看图 7-32 中的代码，会发现 userSession 这样的设置，如图 7-33 所示。联系到它下方是 username 和 password 等信息，我们可以猜测出所在函数 web_submit_data() 实际是发送用户登录信息给服务器的函数，因此 userSession 所在的页面应该是系统的登录页面，如前文中图 7-24 所示。但是，userSession 在实际的网页中却看不到。要理解这样的情况首先要了解 Session 是怎么回事。

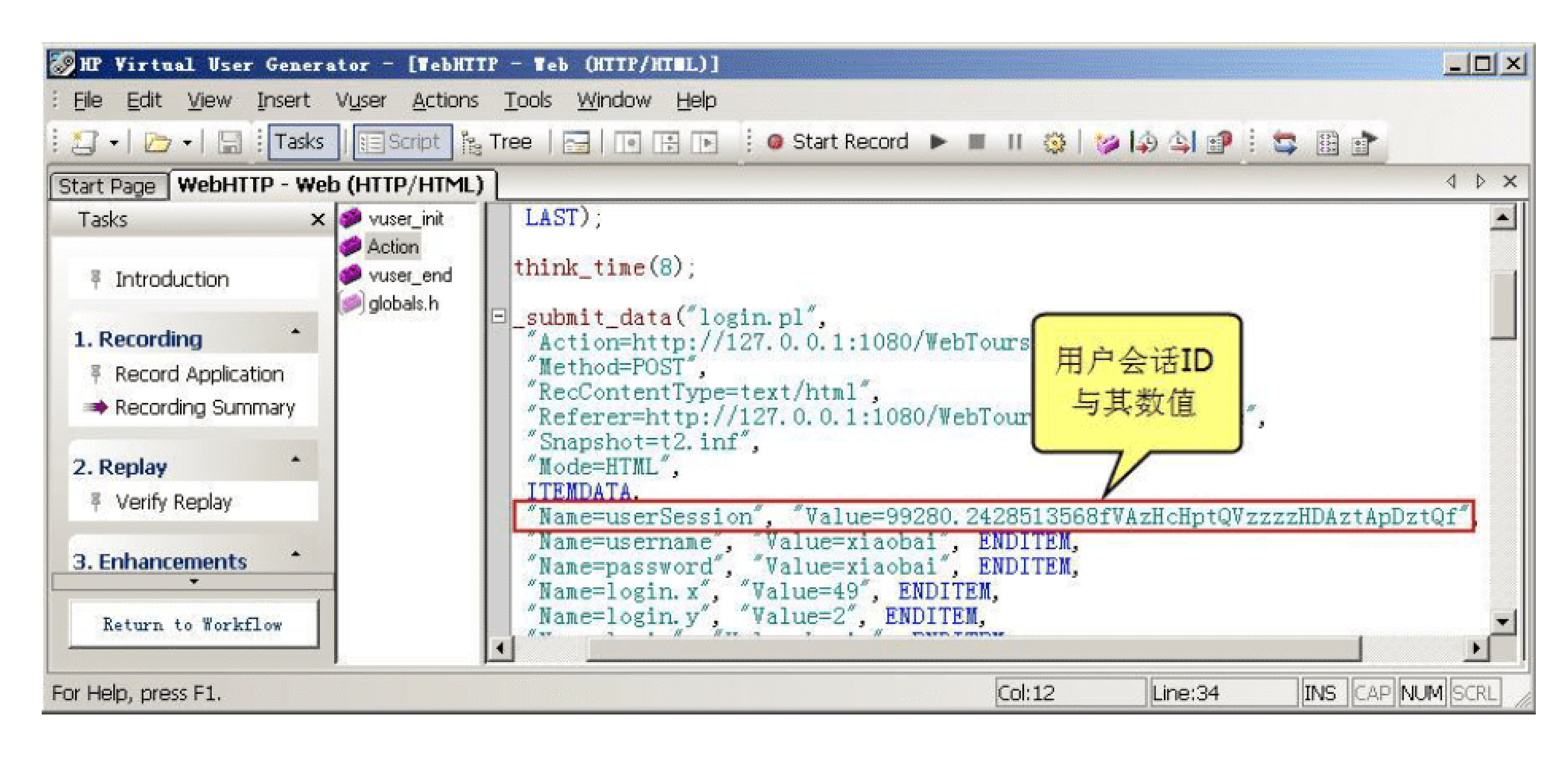


图 7-33　脚本中有关用户会话的 ID 及其数值

【Session 会话】

所谓 Session 就是指有始有终、一系列的操作，比如我们打电话时，拿起听筒、拨号、通话到挂断电话这一系列的动作就构成了一个会话。在 Web 领域，比如我们访问邮箱，从登录账号开始，浏览邮件列表、查看邮件内容、下载附件、回复邮件等，直到退出系统就构成了一个会话。

【有关会话超时】

在实际浏览网站的过程中，我们往往遇到过这样的情况：登录邮箱后，由于有别的事情临时需要去做，很长的时间都没有对邮箱进行任何操作，当我们再回来查看邮件的时候，就会收到系统的错误提示，比如会话超时等。这说明，我们的每一步操作服务器都会记录，如果长时间空闲的话，系统将认为该用户已经退出了网站，为了安全等原因，不再提供完全的服务。如果需要，用户需要重新登录。

【会话的保持】

实际上，在用户每次浏览网站或 Web 应用的时候，绝大多数系统都会分配给该用户一

个会话号码（SessionID），就好比银行中分配给办业务的储户一个号码一样。系统通过不同的号码辨别不同的用户，为他们提供服务。但是，在会话超时之后，如果用户想凭借之前的会话号码重新进入是不行的，必须重新申请新的号码，也就是重新登录。这种情况与错过银行的叫号柜台会拒绝办理是一个道理，要想获得柜台的服务，只能重新领号。

了解了以上的一些背景知识，就可以判断出 7.3.7 节小白播放录制脚本的失败原因：

由于录制时用户 xiaobai 登录时产生了一个 SessionID（在订票系统中被命名为 userSession，其具体命名实际并无限制），录制结束后该会话失效。当播放时，如果还用录制时候的会话 ID，肯定会被系统拒绝，导致无法登录，自然也就找不到登录后才会出现的 Flights 按钮。

原因找到了，下面介绍一下解决的办法。那就是利用 LoadRunner 提供的 Correlation（关联）命令解决会话 ID 变化的问题，将在 7.3.9 节进行讲解。

7.3.9　创建 VuGen 脚本 VII：利用关联解决脚本播放失败

关联（Correlation）是 LoadRunner 中有关脚本处理的一个常用功能，应用在如下的场合：

当客户端（浏览器等）必须根据服务器端此次返回的信息来构成下一次发送回服务器端的数据时，可以利用关联在此次数据中找到它们，再根据预先设置好的规则转换形成新的请求数据，进行下一次的发送。

举前文的例子来解释上述较长的过程：会话 ID（在订票系统中是 userSession）在用户浏览登录页面时由服务器端返回（即此次服务器端返回信息），以隐藏域（hidden field）的形式存在于网页中。当用户输入用户名、密码准备发送回服务器端（即下一次发送会服务器端）时，这个会话 ID 也会一并发送出去。

1．关联操作的步骤

对脚本增加关联操作的步骤如下：

（1）从服务器端返回的数据中选取需要进行关联操作的数据。

（2）将该数据以脚本参数的形式存放，避免在脚本中成为硬编码（Hard code）。

（3）将脚本中需要使用该数据的地方都用第 2 步创建的参数进行替代。

2．硬编码的问题

首先介绍一下什么是硬编码。

【硬编码 Hard code】

所谓硬编码，英文原文是 Hardcode 或者 Hard code，俗称就是“写死”。它是指在代码中容易发生变化的量直接用一个值而不是数学符号来表示。与 Web 应用相关、在代码中“写死”的常见情况有：服务器名称、数据库名称、连接用户名、密码、某些常量的值等。

硬编码在实际工作中应该尽力避免，它们应该用数学符号表示的参数，而不是实际的数值。这样做有两个好处：

- 数值不如字母等数学符号能包含更多的意义。假设一个 Web 应用的最大并发用户数是 5000，那么在代码段中，使用一个类似 Max_Concurrent_User 这样的参数肯

定比直接使用 5000 更让人明了，因此也减少了日后代码维护的成本，不容易出错。

- 使用数值的方法不适应代码的修改。假设某一天最大并发用户数量修改为 10000，那我们需要在成千上万行代码中找到 5000 这个数值，进行修改，如果恰好有别的表示不同含义的 5000，则还要避免错误修改，从而造成很大麻烦。在这样的情况下，可以使用配置文件之类，在其中令 Max_Concurrent_User=10000，而代码中只需要引用 Max_Concurrent_User 即可。这也是业内一贯使用的方法。所谓硬编码，也就是指“写死”的情况给人感觉非常硬，不能快速变化。

3．对脚本进行关联

小白在了解了以上的知识之后，就开始对录制成功的脚本进行关联操作。在 LoadRunner 中，关联有两种，分别是自动关联和手工关联。前者通过菜单、对话框的设置进行，具备方便快捷的特点，但不很灵活，也只能适用于少数的几个协议（不过，对于本书讲述的 Web 应用常用协议，自动关联是可以使用的）；手工关联通过在代码编辑窗口调用关联函数进行设置，具备灵活性，是 LoadRunner 高手必备的方法。不过，在本章，小白需要先学习自动关联的方法，手工关联方法将在之后讲述 LoadRunner 脚本编写的章节中进行讲解。

（1）单击虚拟用户生成器 VuGen（即 LoadRunner 录制脚本的窗口）的 Vuser 菜单，下拉菜单项中选择 Scan Script for Correlations（扫描脚本以发现关联）命令，如图 7-34 中阴影所示。

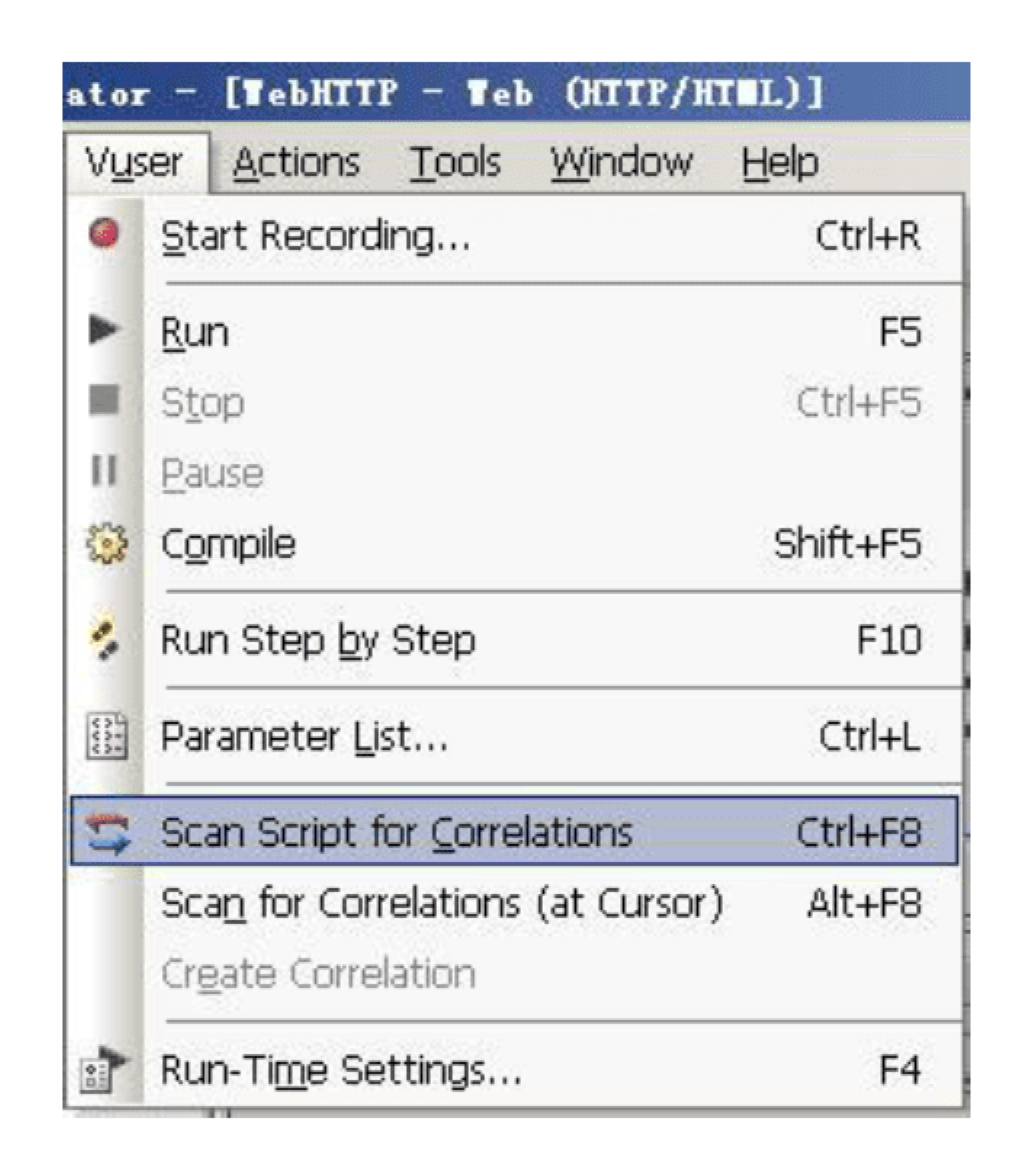


图 7-34　Vuser 菜单项中的扫描脚本以发现关联菜单项

（2）单击该菜单项后，LoadRunner 将扫描录制脚本和播放时的录制脚本，从中发现共同点，并将结果高亮显示在窗口内，如图 7-35 所示。

（3）在图 7-35 的右下方，选中列出的可供关联的一行结果，在用红线标出的位置上单击第一个 Correlate（关联）按钮，即可将被选中的一对数值（在图 7-35 中，该对数值的前者为录制时用户的会话 ID，后者为播放时用户的会话 ID）进行关联。

（4）实现关联后，刚才单击过的 Correlate（关联）按钮将自动变为 Remove Correlate

（去除关联），这样，该按钮就可以对脚本中的现有关联进行管理。

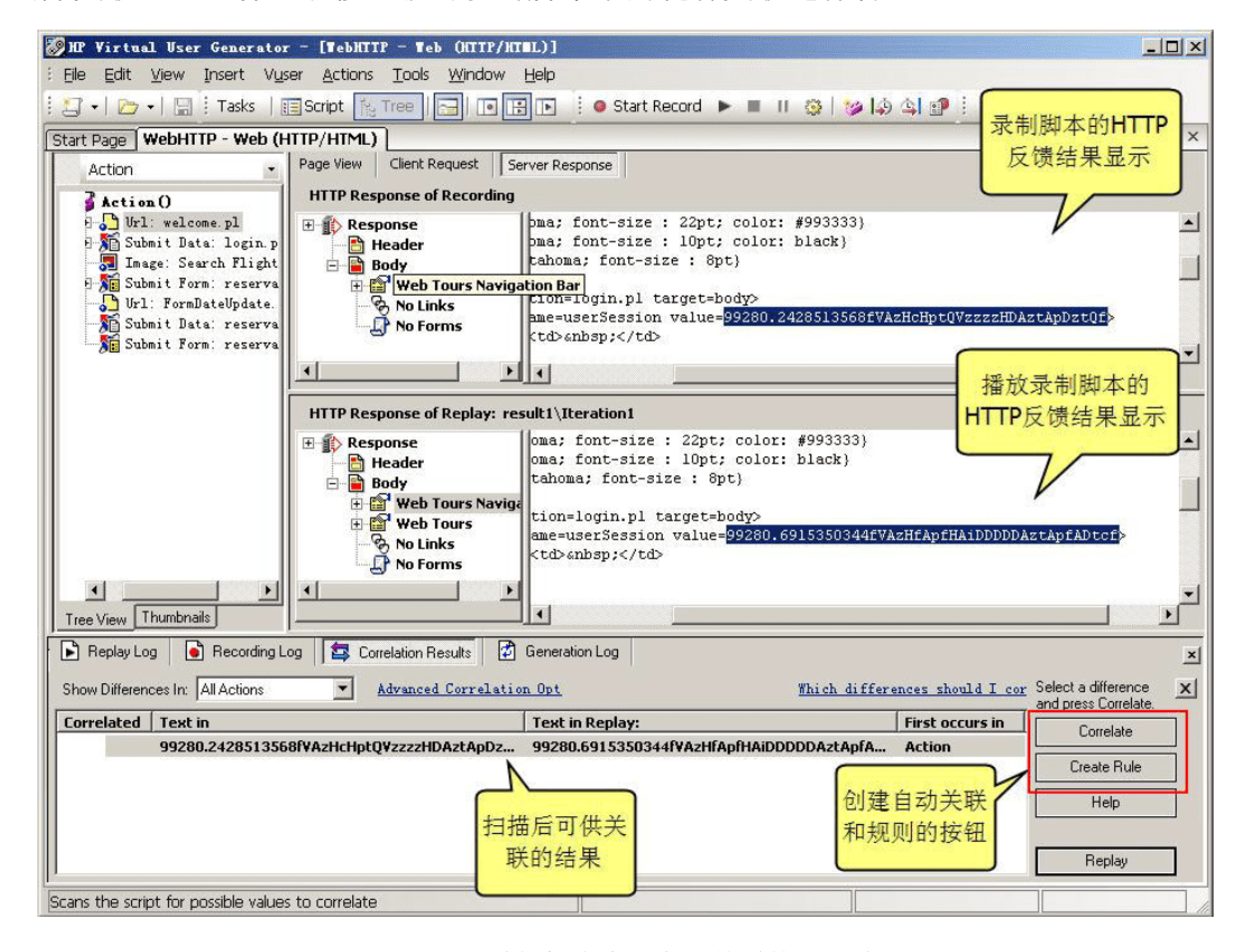


图 7-35　通过扫描脚本以发现关联的运行结果

【Script 界面与 Tree 界面】

有不少初学者在创建关联后无法回到录制脚本的编辑界面，在这里特别指出快速的方法。图 7-35 的结果处于树形（Tree）界面，在导航工具条中可以看出 Tree 按钮为灰色的状态。如果需要切换回脚本编辑界面，只需要单击 Tree 按钮旁边的 Script（脚本）按钮即可。导航工具条有关这两个视图操作的放大示意图如图 7-36 所示。

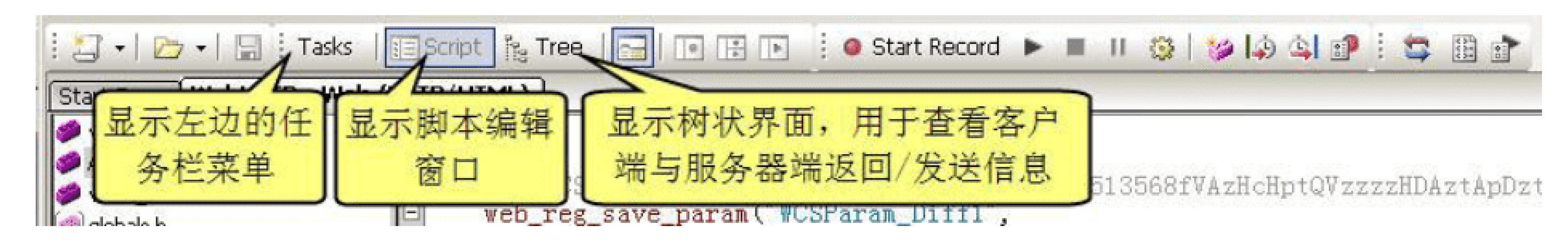


图 7-36　控制显示视图的导航工具按钮

结束自动关联操作后，通过单击图 7-36 中的 Script 按钮切换回脚本编辑视图。可以发现，之前录制的脚本中 userSession 的值为很长的一串，现在已经修改为一个参数，如图 7-37 所示。

现在小白可以直接单击导航工具条中的播放按钮，录制脚本就可以一直运行，而没有在图 7-37 下方的日志窗口出现任何错误。如果要打开界面左边的导航条（通过图 7-36 左上方单击 Tasks（任务按钮显示和隐藏），也可以通过单击 Verify Replay（验证播放）链接实现播放，不同的是，播放成功后会有播放总结页面，如图 7-38 所示。

至此，小白在 7.3.7 节中遇到的问题得以解决。

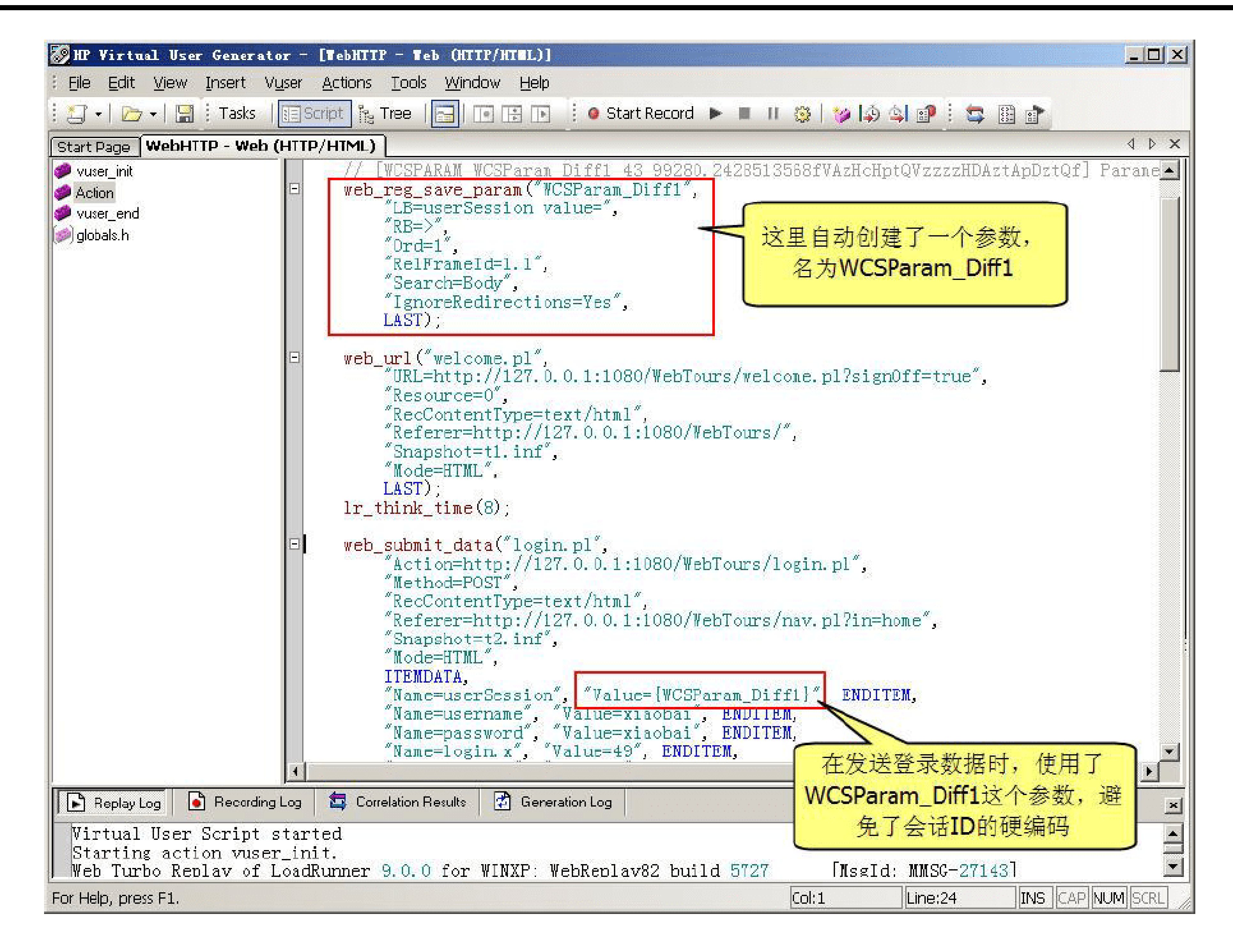


图 7-37　设置自动关联后的录制脚本发生变化

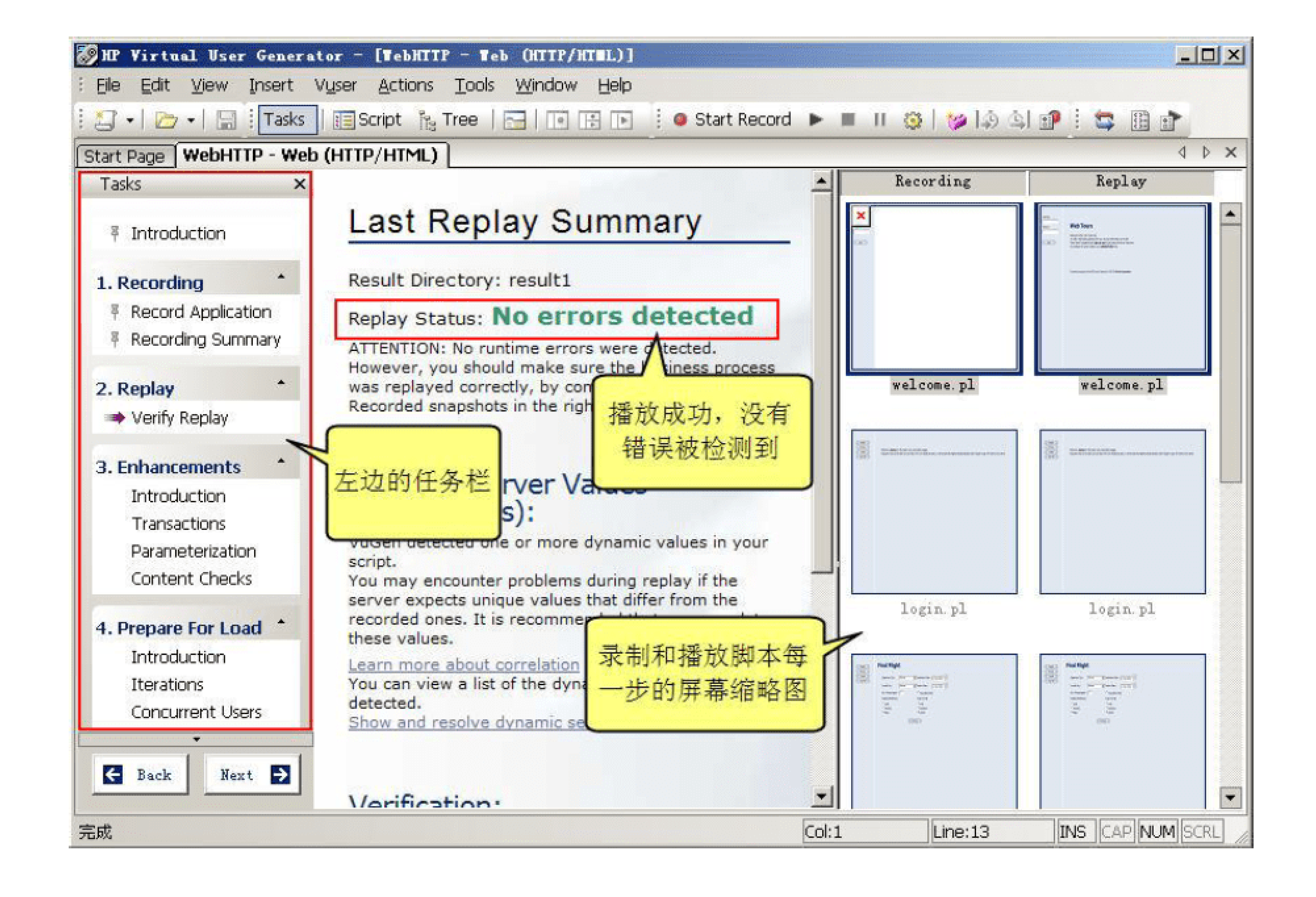


图 7-38　显示播放成功的总结页面

4．保存和导出/导入脚本

小白终于有了自己的实验成果，还成功地解决了实际问题，他首先想到的是将脚本保存，有可能的话分发给经理或者其他同事也来尝试。在 LoadRunner 中保存脚本与其他软件的保存文件没有什么不同，依次选择“文件”|“保存”（File|Save）命令，或者直接按下

Ctrl+S 键即可。分发脚本则有特别之处：在 LoadRunner 中，脚本可以用打包文件的方式进行分发和共享，操作方法如图 7-39 所示。

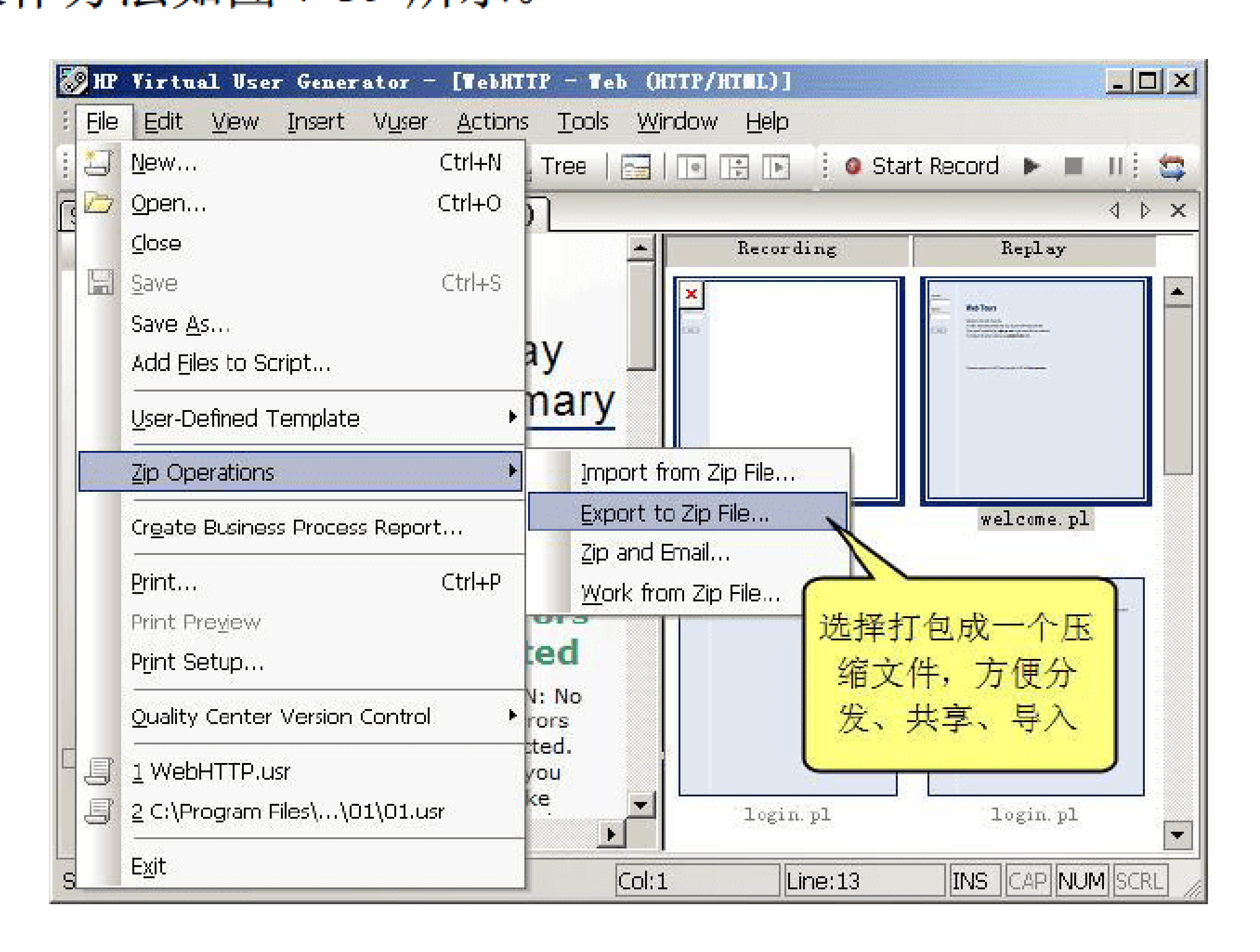


图 7-39　将脚本导出为一个 Zip 压缩文件方便分发、共享和导入

在图 7-39 所示的菜单上选择 Export to Zip File（导出到 Zip 文件）选项后，LoadRunner 会提示输入 Zip 压缩文件保存的位置，根据需要找到后单击“确定”按钮即可。在得到别人提供的脚本压缩 Zip 文件后，也可以通过图 7-39 中的从 Zip 文件导入菜单将脚本加载到本机的 LoadRunner 中。

本节介绍的关联方法是 7.3.8 节末尾提到的，解决会话 ID 不同导致播放失败问题的一种方法，在 7.3.10 节中将介绍利用 Web（Click and Script）协议进行录制。

7.3.10　创建 VuGen 脚本 VIII：利用其他 Web 协议进行录制简介

对于录制支持的 Web 协议，前文提到过还有另外的选择，那就是 Web（Click and Script）协议。将 7.3.9 节中生成的脚本保存并关闭，选择 File | New 命令，在弹出的协议选择对话框中清除默认选择，改为 Web（Click and Script）协议，如图 7-40 所示。

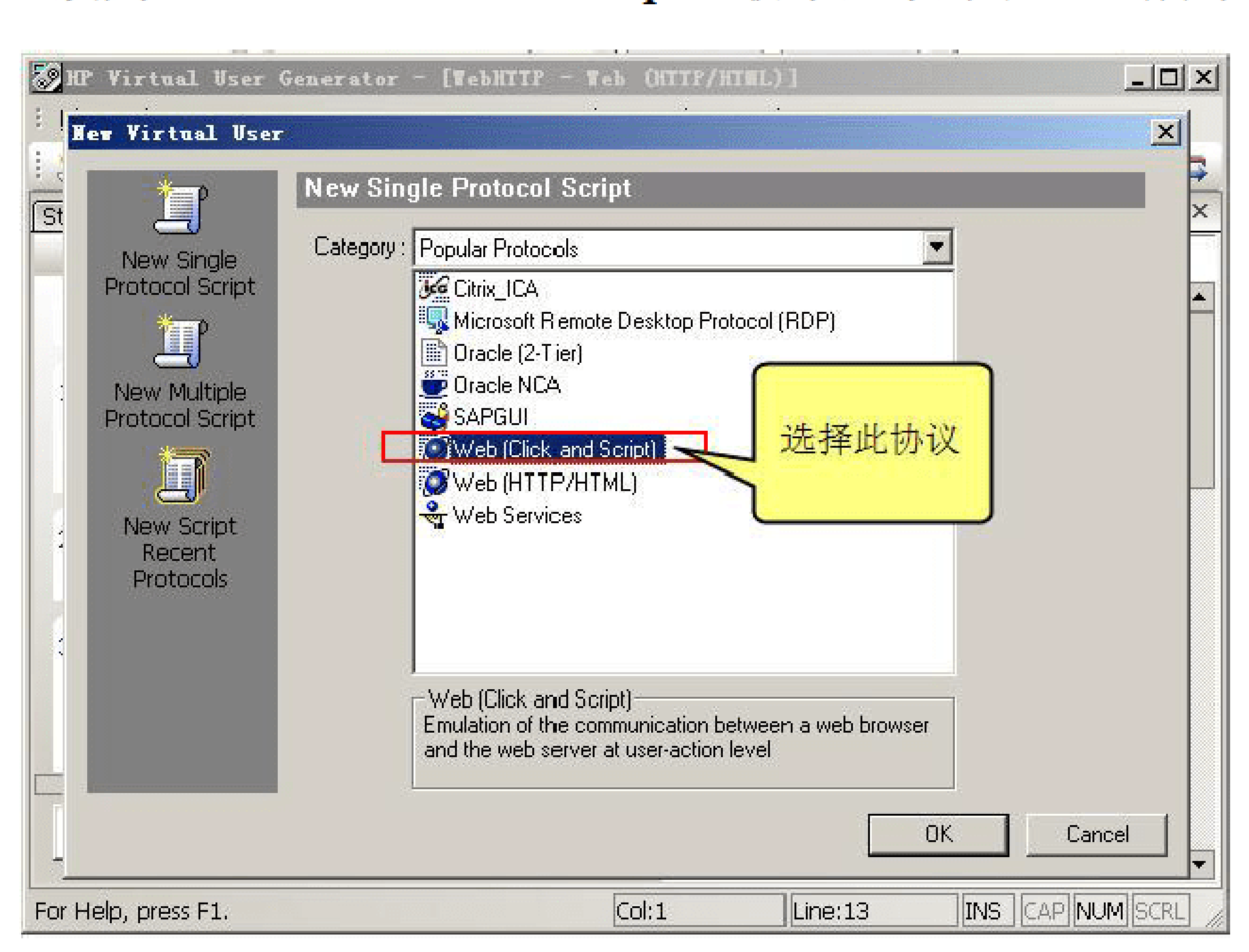


图 7-40　修改录制协议

单击 OK 按钮后，LoadRunner 将保留上次成功录制时的数值，并显示在如图 7-41 的开始录制设置界面，但小白还需要确认录制的脚本是基于 GUI（为 Web（Click and Script）协议默认，但由于之前录制过脚本，可能已经修改为其他形式的脚本）的，为此还需要单击图 7-41 左下方的 Options（选项）按钮。

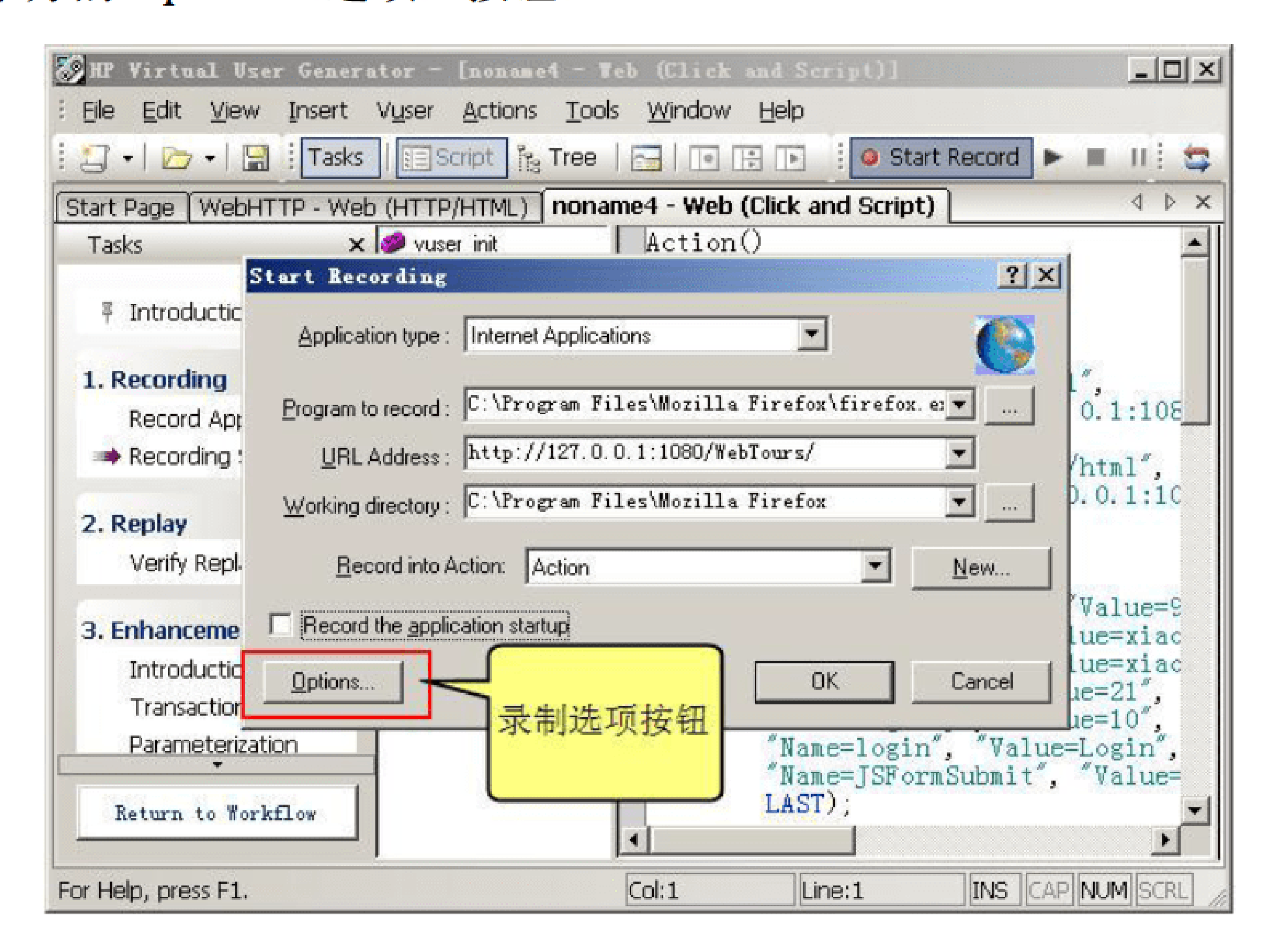


图 7-41　设置录制选项

单击按钮后，确认脚本类型为基于 GUI，如图 7-42 所示。也可以直接单击图 7-42 下方的 Use Defaults（使用默认）按钮，将所有选项都置为默认设置，以快速消除之前录制的影响。

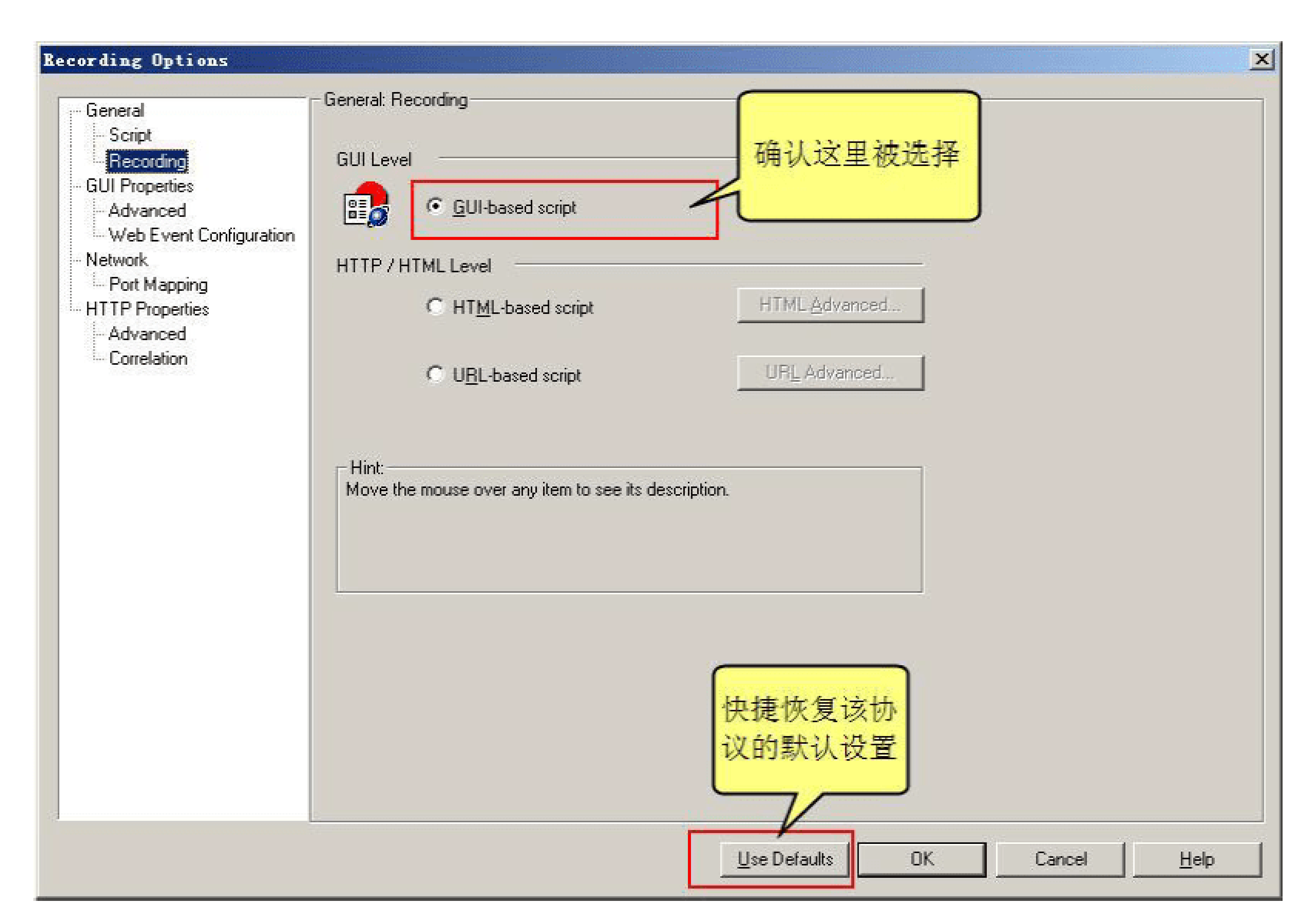


图 7-42　确认 Web（Click and Script）协议的脚本类型

在确认设置正确之后，就可以进行录制了。小白单击了图 7-41 中的 OK 按钮，在打开订票网页后开始录制。

结果却令人沮丧，结果空空如也。经过查询 LoadRunner 的帮助文件，小白发现了这两种 Web 协议的区别。

【两种 Web 录制协议的区别】

两种 Web 录制协议的区别在于：

如果 Web 应用中包含 VBScript 和 Java 小程序（Applet），或者访问 Web 应用的客户端不是浏览器，而是专门开发的客户端程序，则采用 Web（HTTP/HTML）协议录制，它可以提供两种类型的脚本（基于 HTML 和基于 URL）。这也是订票网站所需要采取的协议，因为在选择日期的网页中应用了 Java 小程序。

如果 Web 应用属于其他的情况，或者包含 JavaScript，就采用 Web（Click and Script）协议进行录制，它可以录制 3 种类型的脚本（基于 GUI、基于 HTML 和基于 URL）

【更多的录制协议】

实际上，在图 7-40 中，只列出了常用的协议，可以通过图 7-40 的 Category（类别）下拉列表框选择更多其他的协议。同时，还可以创建使用多个协议的脚本，方法是单击图 7-40 左边图标栏中的 New Multiple Protocol Script（新多协议脚本）按钮。通过所支持的丰富协议，以及多种协议的协同工作，使得 LoadRunner 能够适应相当广泛的性能测试要求。

7.3.11　LoadRunner 进行性能测试的简要步骤

在本章学习了 LoadRunner 录制脚本等入门知识之后，小白对于该软件已经有些熟悉。最重要的是，结合帮助文档与实践，他领悟到 LoadRunner 性能测试的几个步骤，如图 7-43 所示。

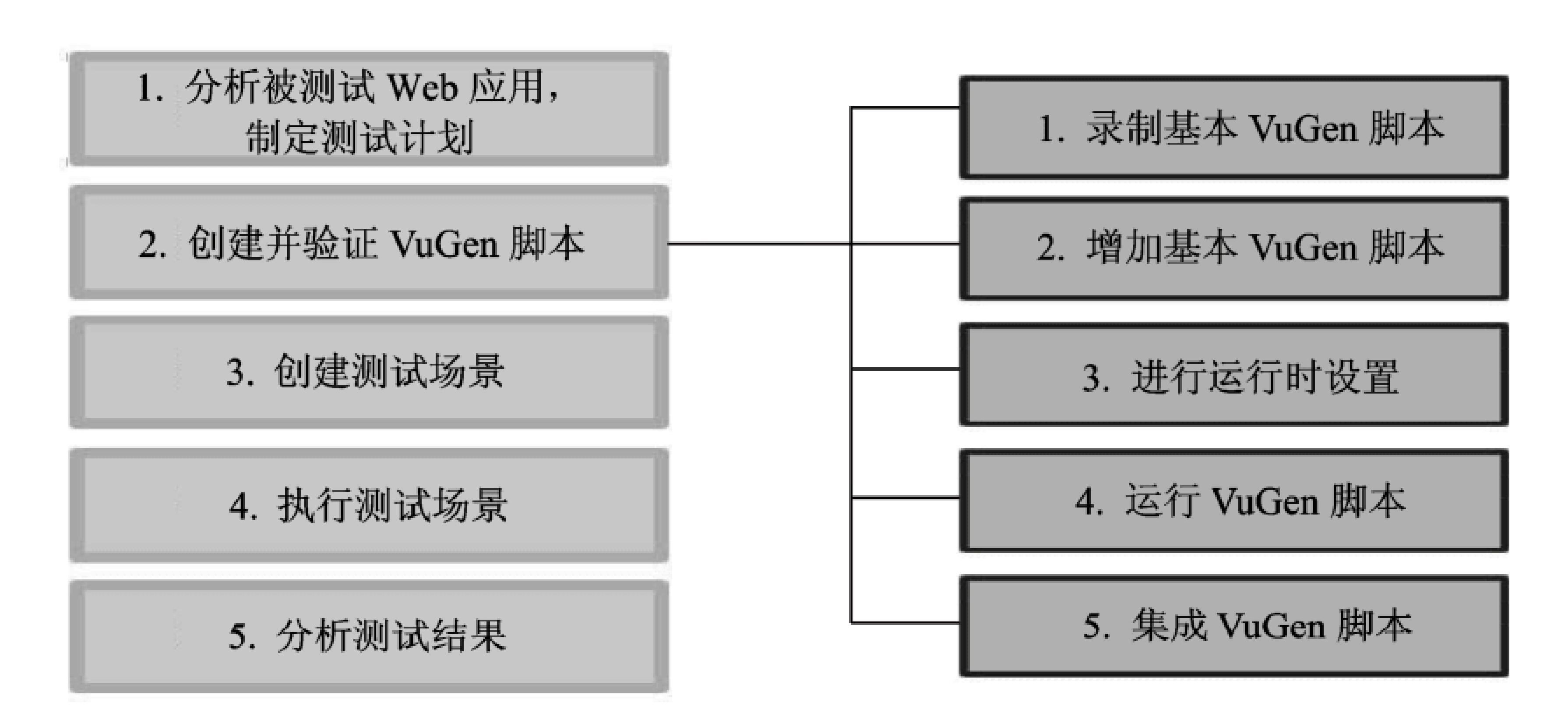


图 7-43　LoadRunner 进行性能测试的步骤

截至目前为止，小白已经对图 7-43 中的左边第 2 步以及其包含的右边 5 个具体步骤比较熟悉。在今后的几章中，在学习性能测试理论的同时，他会不时地进一步学习 LoadRunner，尽快熟悉其他几个步骤的使用方法和技巧。

7.4　本 章 小 结

在本章，小白受经理的要求，对性能测试软件进行了一番选型，并提出了自己的建议。

对于选择性能测试软件来说，要从如下几个方面来考虑是否符合所在公司和组织的要求：

- 功能上是否符合被测试 Web 应用的要求。
- 成本上是否符合公司的要求。特别注意不同的软件版本、许可方式对成本的影响。
- 技术支持是否相对迅速、业内使用人数是否较多等其他因素。

经过选择，小白所在公司决定使用 LoadRunner 作为性能测试的工具软件。为了尽快进入状态，小白下载了试用版，并对 VuGen 的脚本录制进行了实际操作。创建 VuGen 脚本的步骤如下。

- 根据不同协议录制基本 VuGen 脚本。注意 Web（HTTP/HTML）和 Web（Click and Script）的区别。
- 在脚本中增加交易与关联增强基本的 VuGen 脚本。特别注意关联对于会话 ID 不同导致播放失败问题的解决。
- 对运行时的脚本进行设置，主要分为两项：迭代次数，代表同一操作循环的次数；并发用户：同时请求 Web 应用的用户数量，与场景密切相关。
- 验证、运行和集成 VuGen 脚本。

通过动手实验，小白逐渐熟悉录制脚本的操作，也总结出 LoadRunner 进行 Web 应用性能测试的主要步骤，它们是：

（1）分析被测试的 Web 应用，制定测试计划。
（2）创建并验证 VuGen 脚本。
（3）创建测试场景。
（4）执行测试场景。
（5）分析测试结果。

从上述步骤可以看出测试计划的重要性。在实际工作中，测试计划的编写绝不限于使用 LoadRunner 进行性能测试之前，而是在几乎所有的性能测试、甚至其他测试开始之前。在第 8 章中，读者将和小白一起学习如何编写测试计划，以及测试计划都有哪些要点。

第 8 章　编写测试计划

小白所在公司的网站已经进行了开发的后期阶段，此时，他经过前述几章的学习，对于工作中所需要使用的性能测试软件 LoadRunner 也有了一定的了解。但是，对于整个性能测试、乃至测试过程而言，只了解工具的使用是不够的。总体来说，整个性能测试的过程需要包括更多的内容，主要步骤如图 8-1 所示。

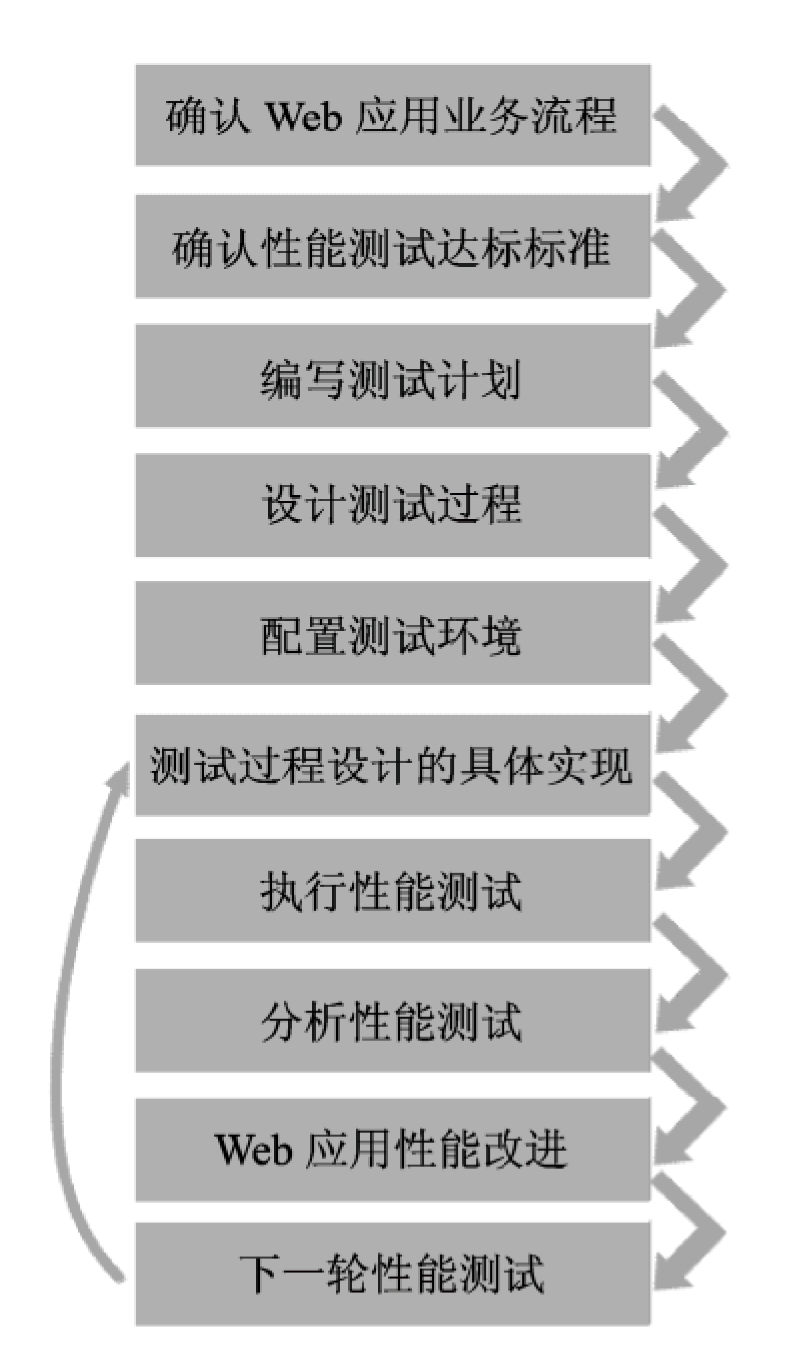


图 8-1　性能测试的主要过程

对于小白在前面几章中所学到的 LoadRunner 录制、播放脚本、用户场景设置等技术，可以归入图 8-1 中的设计测试过程或测试过程设计的具体实现这两个阶段。在本书中提前讲解是为了使得读者能够提前熟悉性能测试的具体内容。

图 8-1 非常重要，性能测试工程师在实际工作中应当清楚地明了自己当前工作所处的阶段与目标。

本章主要介绍编写测试计划，这个步骤一定要在实际开展工作之前完成。为了写出一个好的测试计划，首先需要对 Web 应用的业务流程有所了解，找到性能测试的侧重点，并确定性能测试达标标准。一般来说，以上的这些信息都将列入测试计划。

8.1　了解被测试 Web 应用的结构

当我们拿到一个 Web 应用进行性能测试之前，首先需要了解它的 3 种类型的结构：逻

辑结构、物理结构和系统结构。这 3 种结构能够加深我们对于 Web 应用的整体把握，熟悉与性能相关有哪些软硬件部分。

对于被测试 Web 应用理解不深入会对日后的性能测试工作造成不利的影响：测试工程师可能对于哪个部分会严重影响性能知之甚少，从而不能有针对性地进行性能测试和性能优化。了解上述 3 种结构的方法很简单：

- ❑ 花尽量多的时间阅读软件说明文档。
- ❑ 花尽量多的时间与相关人员（网络管理员、软件架构师、测试经理等）沟通，了解最终 Web 应用的运行环境（业内一般称之为“生产”环境）和测试环境。

首先，我们要学习 Web 应用的逻辑结构。

8.1.1　逻辑结构

所谓 Web 应用的逻辑结构，通俗地说，就是从整个系统抽象出来的几大部分，它与具体的软件实现、硬件并没有关系。

Web 应用常见的逻辑结构就是“3 层结构”，分别如下。

（1）客户层：就是终端用户的电脑。用户通过它访问 Web 应用。

（2）表现层：Web 应用的界面以及后台的功能实现。表现层处理所有的商业逻辑（订单处理、用户注册等），并负责向客户层派发数据。

（3）数据层：Web 应用的后台数据库，存储系统所需要的各种信息。

逻辑结构往往在各类会议中应用较为广泛，因为它脱离具体的实现细节，易于理解。图 8-2 显示的是小白所在公司网站的三层结构示意图，图中箭头表示数据的流向，双箭头表示数据的传递方向是可以相反的。

【三层结构就是 3 台电脑吗】

有的读者看到图 8-2 后可能感觉三层结构就是 3 台不同用途的电脑（或者服务器），但实际并不是这样的。三层结构是一种逻辑上的结构，其中某一层可以包括 1 台服务器，也可以包括多台服务器。甚至，两层甚至所有的三层都可以共存于一台服务器之上。下面就是一个极端的例子：假设公司的 Web 应用总共只有一台服务器，Web 服务和数据库都处于这台服务器上，小白为了方便，直接通过该服务器上的浏览器访问该应用。在这样的情况下，三层就共处于一台服务器上。所以说，三层结构或者逻辑结构反映的是逻辑上 Web 应用的几大部分，与软件、硬件的具体实现没有很大联系。

在实际工作中，大中型网站的逻辑结构往往比较复杂，不会都如图 8-2 所示的那样简单。图 8-3 列出了关于某 Web 应用的一个较复杂的逻辑结构图。

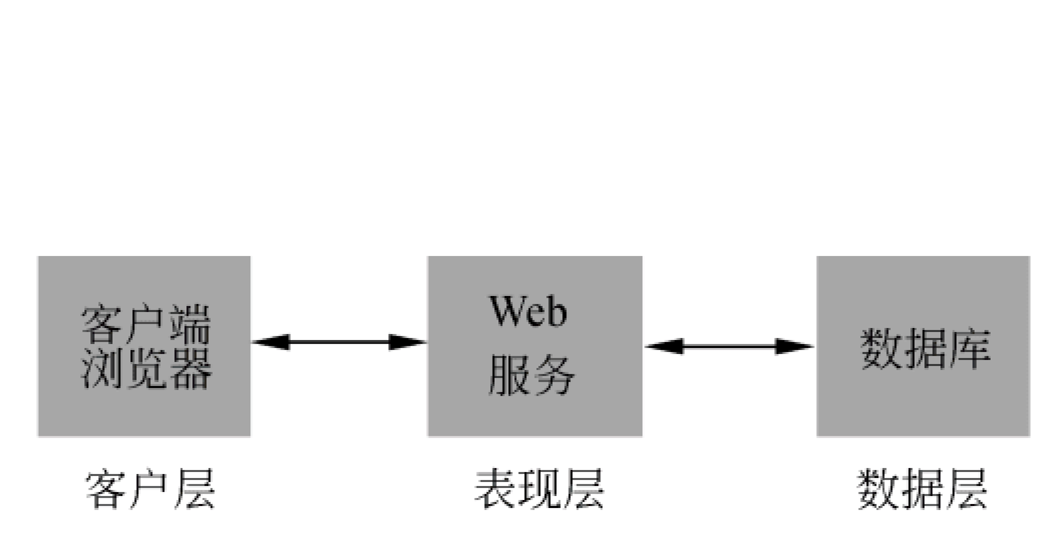


图 8-2　一般的三层结构示意图

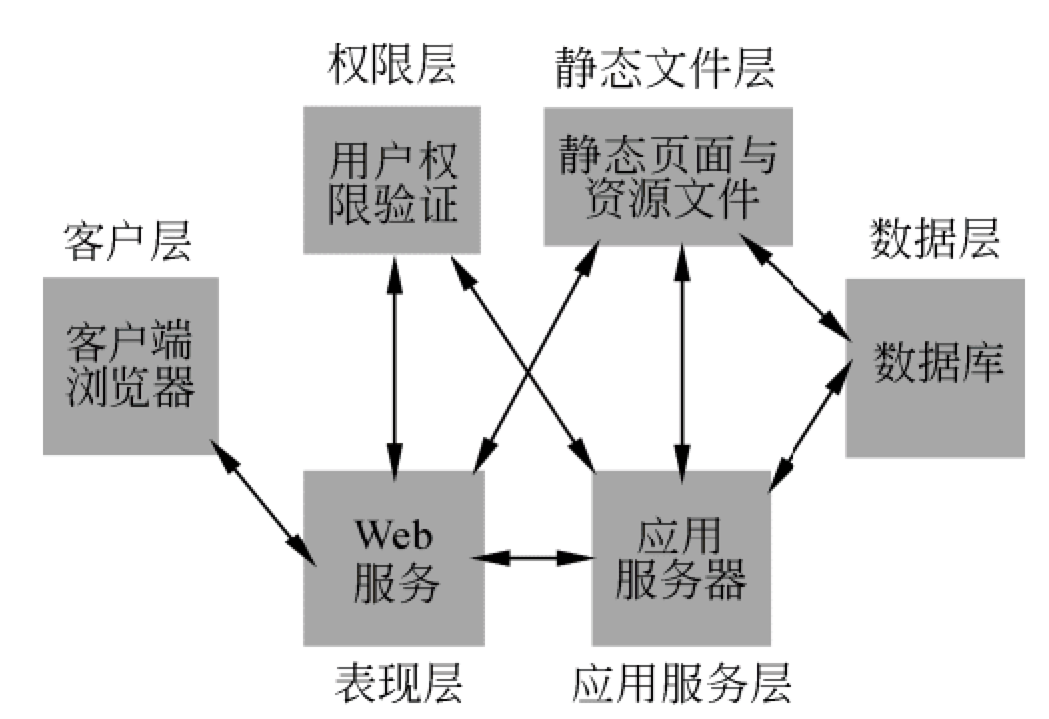


图 8-3　一个复杂的 Web 应用逻辑结构示意

在图 8-3 中，我们可以看到表现层被细分为表现层（Web 服务）、权限层（主管用户验证等功能）、静态文件层（存放在文件系统中的静态 Html 页面、图片、音视频等文件）、应用服务层（如 Weblogic 等应用服务器）等几个部分。实际上，对于客户层和数据层，一般不会有更细的划分出现。

【逻辑结构由谁决定】

对于 Web 应用的逻辑结构，一般都由技术总监、软件架构师给出，不需要性能测试工程师自行归纳总结。但是，工程师在实际工作中需要对整体结构中的各大部分烂熟于心。

8.1.2　物理结构

物理结构指的就是构成 Web 应用的硬件结构。性能测试工程师可以从网络管理员处获得关于 Web 应用的物理结构图。

业内有时候也把物理结构图称为拓扑图，或者简称为 Topology。图 8-4 是某 Web 应用的物理结构示意图，在其中，我们可以发现防火墙、服务器、数据库服务器、负载均衡器、网络加速器等多种硬件设备。各个硬件的连接线则代表了网络上的互通性。

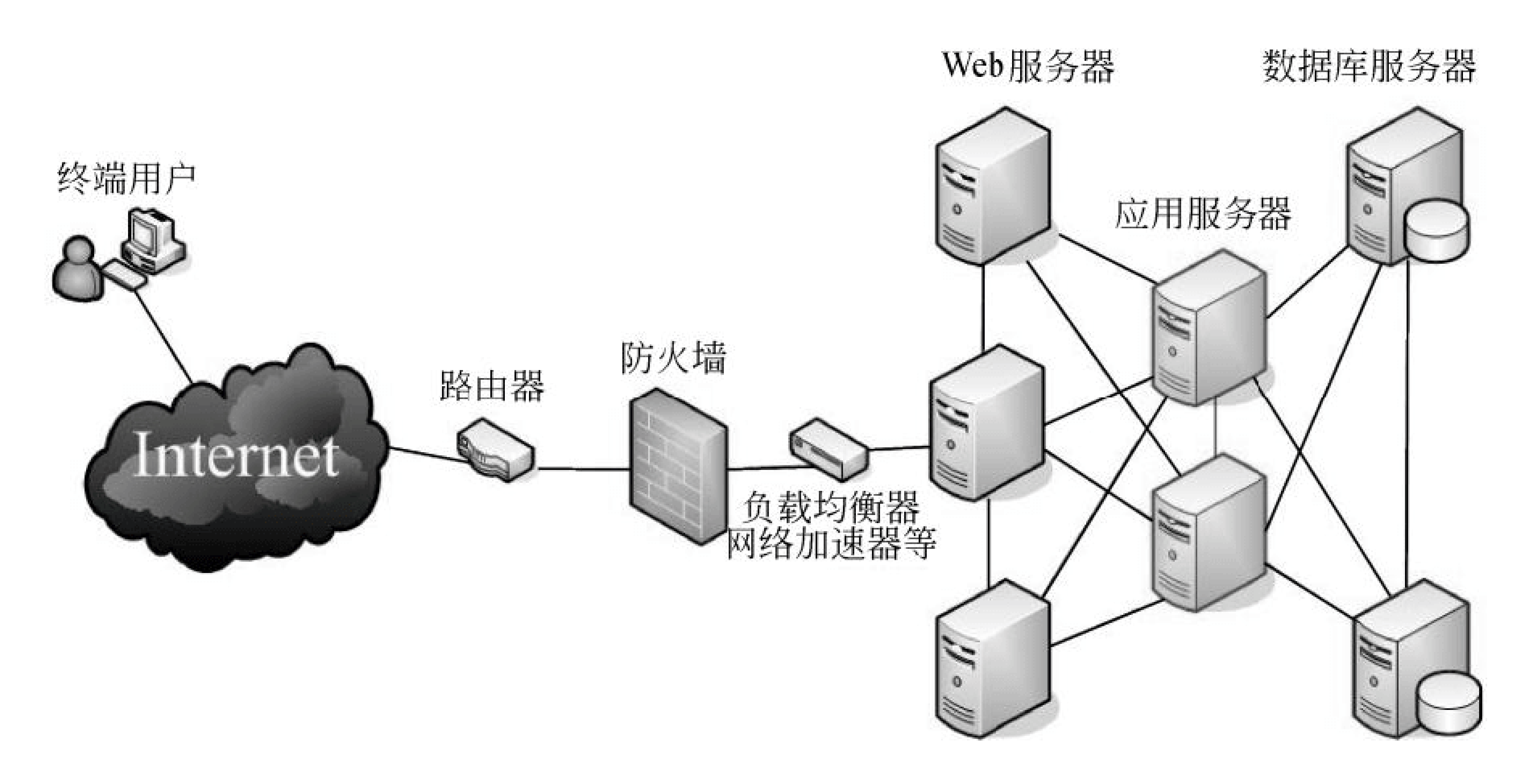


图 8-4　一个物理结构示意图

了解 Web 应用的物理结构图，有利于性能测试工程师判断由于硬件方面造成的性能问题出现的大致位置，还可以依据对结构的了解有针对性地设计测试用例。

8.1.3　系统结构

从逻辑结构和物理结构出发，将两者相结合，就能够得到系统结构。性能测试工程师需要特别关注系统结构中与性能有关的部分。图 8-5 是依据图 8-4 画出的系统结构图。

读者可以对照图 8-4 和图 8-5，其实二者非常类似，只是具体的服务器图标用逻辑结构图中的方块来表示了，这样看起来更加直观。

【3 种结构图的不同】

逻辑结构图中一个方块可以代表多台服务器，或者多台服务器上的不同程序。物理结构图中只有具体的硬件表示。而系统结构图中用逻辑结构图的方块等抽象形状来代表具体

的服务器。这就是 3 种结构图的主要区别。

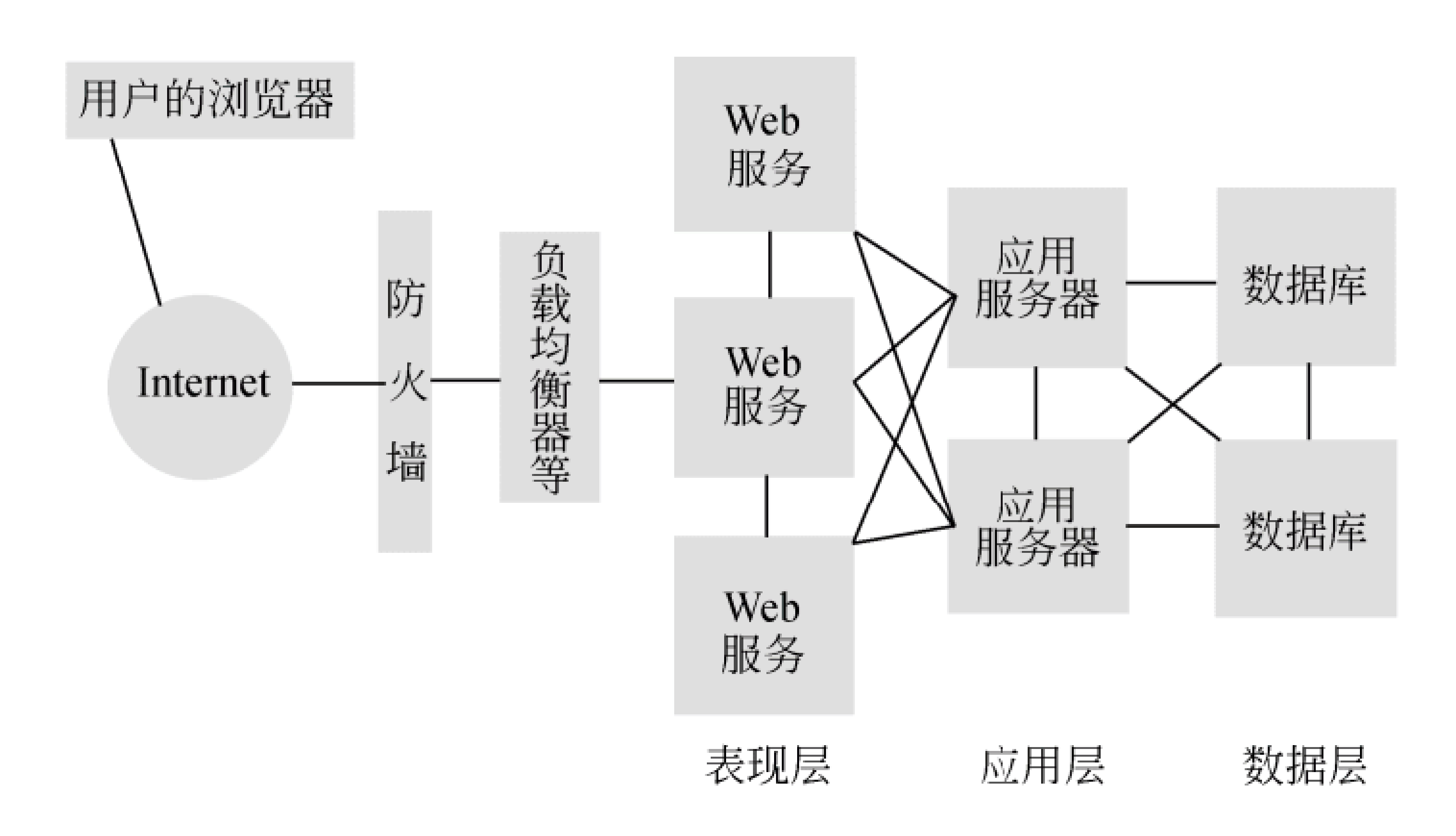


图 8-5　某 Web 应用的系统结构图

逻辑结构图、物理结构图和系统结构图实际上都可以代表当前 Web 应用的结构，它们是应不同背景、不同岗位的人员了解和讲述方便而产生的。对于性能测试工程师，可以在这 3 种结构图上标出自己感兴趣的部分，比如，哪些可能是性能瓶颈，哪些可能产生偶然问题而导致性能变差（例如网络不通畅、磁盘空间满、电源问题等）。

8.2　确认业务流程

小白已经能够使用 LoadRunner 进行脚本的录制了，他自己认为，如果再将第 7 章中未讲解的场景部分掌握熟练，就应该能够胜任性能测试的工作。但经理仿佛看出了他的想法，对他说："光会使用 LoadRunner 等性能测试软件，并不能说明就掌握了性能测试的方法。软件只是实现想法的工具，重要的还是进一步了解咱们公司网站的业务流程，发掘性能测试要测试的内容。"

那么，性能测试对于 Web 应用来说，要关注哪些方面呢？这就不能不从小白公司的网站说起。

8.2.1　业务流程对性能测试的影响

读者可能与小白有同样的想法，认为学习性能测试就是学习 LoadRunner 如何使用。其实不然，性能测试工具软件的学习只是其中很小、很容易的一部分，在它之前、之后都包含了很多的内容。

曾经有一个比较著名的招聘网站，它使用过类似"看准了你再跳"这样的宣传口号。在本书中，我们可以将其修改为"看准了你再测"。一般而言，对于 Web 应用的功能测试来说，测试工程师需要了解系统中的正常功能，才能通过与实际 Web 应用的行为相比较，判断是否存在功能上的 Bug。性能测试与功能测试在这方面是相同的，都需要发现问题，因此，对于当前 Web 应用在性能上的理想状态，要基本明确。而做到这一点，业务流程的熟悉是必不可少的。

【关键路径】

而熟悉业务流程，很重要的一个目的就是发现其中和性能测试相关的“关键路径”。何为“关键路径”？在本书中并不打算用比较复杂的数学理论来说明，而是通过一个小白邮寄包裹的生活实例来进行通俗意义上的讲解。读者如果感兴趣，可以进一步查看专业的理论书籍获得严格的定义。

【实战演练：关键路径的通俗举例】

假设小白在北京要给上海的朋友邮寄一个包裹，那么该包裹一般情况下，需要经历小白从公司或者宿舍将其带到邮局、在邮局打包、工作人员通过各种交通工具投递到上海、上海邮局通知朋友，朋友去领取等一系列的活动，如图 8-6 所示。如果我们考察整个邮寄过程所花费的时间，由于包裹在路途上的时间（图中为 28 小时，过程总时间为 30 小时）一般占据主要部分，所以如果能减少这部分所花费的时间，就能显著地缩短包裹到达的时间；而由于其他部分花费的时间（比如邮局打包的时间、小白和朋友分别走到邮局的时间）相对固定，减少它们并不能起到很明显的效果。因此，我们可以将包裹的投递过程称为整个邮寄过程中的“关键路径”。

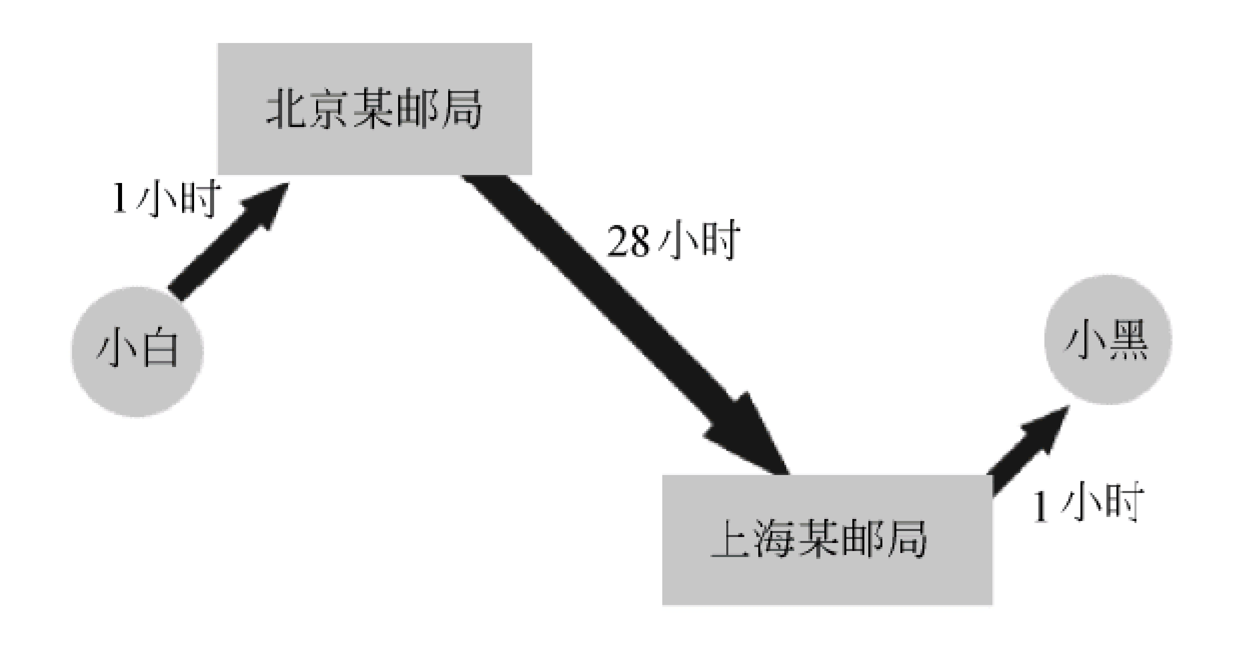


图 8-6　小白邮寄包裹的“关键路径”

从这个例子可以看出，对于整个 Web 应用业务流程的熟悉过程，正是为了能够更好、更快地发现与性能测试相关的关键路径。针对这些路径进行重点的测试，能够使得工作更有效率。

8.2.2　了解 Web 应用的功能模块

小白公司所开发的 Web 应用是一个大型的电子商务网站。通过阅读一些文档，同时，也通过模拟一个普通注册用户的实际操作，小白了解到该网站在功能上可以分为如下几个大的部分，业内人士常常称呼它们为“模块”。

- 用户登录部分。新用户可以进行注册，增加、修改、删除注册用户信息。
- 商品展示部分。包括商品的分类显示、商品的搜索页面以及单件商品的详细信息显示。
- 用户订单部分。包括用户的购物车页面、收藏夹页面、用户资料的修改页面、订单的列表、查询页面等。
- 社区内容部分。包括主题帖列表、用户创建、编辑帖子页面、上传的图片、附件显示功能页面等。

- ❑ 后台管理部分。包括用户信息的维护、社区内容的审核和维护、商品信息的编辑、用户订单的管理页面、优惠活动的设定页面等。
- ❑ 仓库管理部分。包括商品的库存管理、库存商品的调拨与流转、缺货登记、快递公司信息等页面。

如上每一个部分都相对独立，且具备不同的特点，这导致对它们进行性能测试需要额外注意一些问题。从 8.2.3 节开始将一一讲解在这些部分中，性能测试需要关注哪些特别的问题。

8.2.3 确定用户经常使用的功能

在实际测试的准备过程中，在每一个功能模块，都有必要确定用户的关键操作，很类似前文提到的“关键路径”。针对这些关键操作的性能测试结果往往比一般操作的结果更为重要，对性能优化也更有参考价值。

但是，由于网站尚处于开发过程中，我们也不能准确地估计到用户经常使用的功能有哪些。那么，该如何确定它们呢，这方面竞争对手的网站可供参考。通过自己、其他用户对竞争对手网站的访问情况，可以推断出用户的行为。主要有几个途径：

- ❑ 了解竞争对手网站的用户信息。
- ❑ 了解竞争对手网站最终用户的期望行为，比如社区中的意见反馈等。
- ❑ 自己作为一个普通用户，访问竞争对手网站时的感觉。
- ❑ 竞争对手网站的帮助页面、市场宣传手册等。
- ❑ 对于竞争对手网站的新闻报道、第 3 方访问统计等历史数据。

有了这样一些数据作为参照，我们对于当前正在开发的 Web 应用的客户群体就会有所了解，由于用户行为一般来说习惯于一致性，通过多方面的讨论，就能够对原有的估计做出一定的修正。

下面将对 8.2.2 节中列举的网站几大部分与性能测试相关的部分做简单介绍。

8.2.4 用户登录部分与验证码

用户登录部分的功能与目前大多数网站大同小异，值得一提的是，采用了“验证码”技术来避免所谓的“自动登录机器人”等对网站的伤害。

【自动登录机器人】

自动登录机器人是一些程序或者脚本，运行它们可以不通过网站的用户注册、登录等页面，直接发送注册需要的信息（用户名、密码等）给服务器，如果瞬时数量巨大的话，会造成服务器负担过重，影响为正常用户提供服务。另外，这种机器人更重要的用途是进行用户密码的暴力破解。它可以在不知道某用户密码的情况下，进行成千上万次登录尝试，每次发送不同组合的密码，直到服务器返回的信息确认登录成功为止。可见，这种“机器人”对网络安全也很有害。目前，各个网站防止这种现象的方法一般都是每次随机变化的“验证码”图片（也有随机提出问题需要输入字符进行回答的），只有验证码正确，才能进行下一步的操作。由于这类“机器人”读写图片的具体内容尚存在难度，因此能保证登录、注册过程在网页上进行。图 8-7 是网易邮箱注册页面在某一时刻产生的验证码图片。

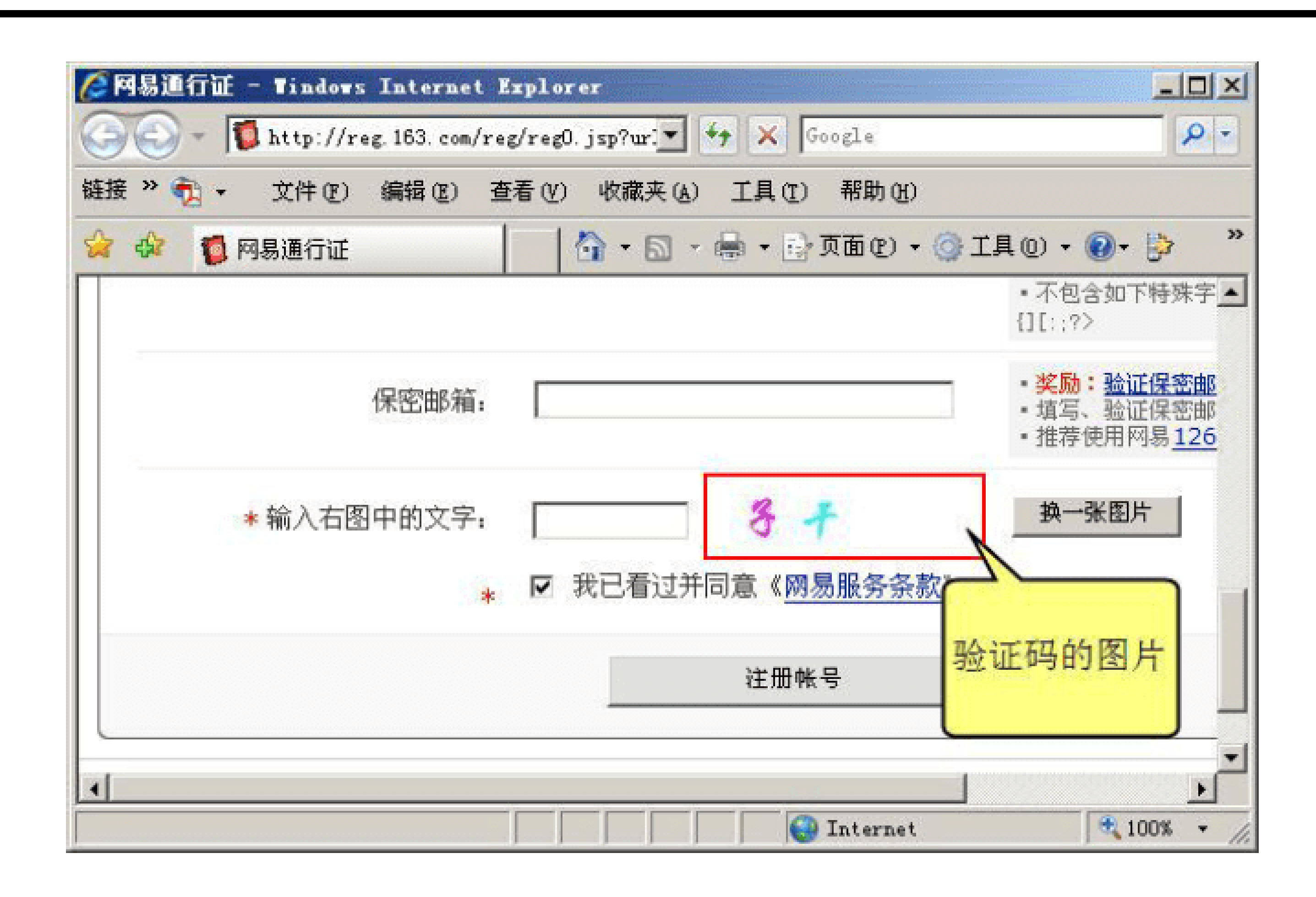


图 8-7　网易邮箱注册页面的验证码图片为汉字

除了自动登录机器人外，还有自动发帖机器人，主要用于网络广告等不良信息的散发。解决办法也类似，在发帖网页上加入验证码图片。

网站通过引入验证码更加安全，但却给自动化的性能测试带来了麻烦（每次登录验证码都不一致，测试工具无法自动读取内容），我们将在后面的章节具体讲解解决这一麻烦的办法。

8.2.5　商品展示部分

用户在使用购物网站的时候，一般来说，用户登录页面只需要访问一次，而主要的浏览量都来自首页、商品列表页面和具体单件商品的展示页面。因此，对于这些页面的访问速度要求会更加严格一些：毕竟，如果一件商品的介绍页面在浏览器中长时间打不开，会影响用户的购物心情。

另外，小白公司网站的商品展示部分利用了第 3 章介绍过的 AJAX 技术，用于商品评论、商品分类切换等处。对于 AJAX 技术的性能测试，也是需要关注的领域。本书将在后面的章节中提到。

8.2.6　用户订单部分

用户订单部分页面主要处理用户生成、提交、修改订单的过程，另外，如果用户选择了网上支付，还要提供银行支付页面的入口和返回页面等。用户订单部分的核心需要 IT 领域中关于事务（Transaction）的概念和知识，因此，对于这部分的性能测试，要重点关注事务的相关性能。在本书第 7 章介绍 LoadRunner 的基本使用段落中，曾经提到了录制后强化脚本这一步骤，它就包含了对于事务或者类似一系列操作的支持。

那么，什么叫做事务呢？请见下文。

8.2.7　事务与网上支付

事务（Transaction）一般与数据库技术相关，是为了满足某个需求而进行的一系列信

息变更操作的总称。事务具有 4 个特点，分别是原子性、一致性、隔离性和持续性。举例来说，我们在银行存钱就可以称为一个事务，它的 4 个特点分别体现在：

- 原子性（Atomicity）：在银行存钱要么成功，要么失败。也就是说，一个事务是不可再分割，这也是原子性这个名称的由来。
- 一致性（Consistency）：这个特性与原子性紧密相关，描述了利用一致性规则保证原子性的方法。举例来讲，在银行存钱之后，无论成功与否，存折中依然还是描述存款的数字，它应该和数据库中其他相关数据对应。
- 隔离性（Isolation）：一个事务的执行不应被其他并发的事务所干扰。举例来讲，银行柜员在处理我们的存款请求时，就不能再为其他请求所占用，这同时也能保证数据的一致性。
- 持久性（Durability）：一个事务如果成功，那么它将不可撤销，系统也无法回复之前的状态。我们在银行成功存款之后，存折上现有金额等状态将一直保持到下次事务之前。

在小白公司所开发的网站上进行网上支付，大致过程如下：

（1）用户选购商品。

（2）收银台结账，于订单生成页面修改送货地址等信息，另外还需要选择银行进行支付。这些信息都会保存在网站的订单数据库中。

（3）网站将金额、订单号码等发送至银行交易网关（一般由各银行提供接口和调用方法）。

（4）银行交易网关进行交易划账处理。

（5）银行交易网关返回成功或者失败信息，网站据此分别返回给用户相应提示页面。

对于上述几点的第（2）步、第（4）步，都是事务的具体体现。图 8-8 显示了在小白公司网站页面上选择网上支付后进入的某银行交易网关。



图 8-8　某银行的网上支付交易网关

8.2.8　社区内容部分

社区内容部分其实类似前文讲述的商品展示部分，对响应时间和可能的 AJAX 应用提出了特别的性能测试要求。另外，与商品展示不同的是，社区一般相对更看重在线用户数量，有些社区网站更会在页面上列出“当前在线人数”等以显示本网站的能力。

8.2.9　后台管理部分

所谓后台管理部分，就是对前台显示的各种信息进行设置和控制的页面。

【网站后台的使用者】

网站后台管理部分的使用者一般是网站内部编辑、图像处理人员等。与此对应，对于在内部网络中使用的 Web 应用，比如各种基于浏览器的办公系统、邮件系统等，虽然使用者是单位内部人员，但他们依然属于终端用户，其使用的界面依然属于前台。在这种 Web 应用下的后台管理部分，指的是系统管理员对整个系统进行设置的页面。

后台管理部分一般来讲，由于使用者局限在单位内部，人数较少，网络条件也好，因此性能一般不成为大问题，但要注意它不能对 Web 应用前台部分的性能产生影响。

- ❑ 后台管理系统程序在物理上应位于不同于网站服务器的其他服务器。这样对于后台管理页面的经常性访问不会占用用户访问网站服务器时的带宽。
- ❑ 后台管理系统程序在功能上不应该对网站性能有所影响。如果由于后台管理程序的 Bug，导致网站数据库失效、性能变差、前台页面出现 Bug，这不单单会干扰我们关注的前台页面性能测试的结果，还在功能上对网站造成了不好的后果。

总而言之，对于 Web 应用的后台管理部分，我们可以只对其进行基本的性能测试，更重要的是要保证它对于其他部分没有坏的影响。

8.2.10　业务流程中有关性能测试的难点

综合前文讲述的网站业务流程中各部分的特点，小白发现了性能测试的若干“关键路径”。但是也发现了目前存在一些完成性能测试方面的难点，列举如下：

- ❑ 验证码变化的问题：需要有一种或几种方法使得自动化性能测试能够适应这种变化。
- ❑ AJAX 技术的性能测试问题。
- ❑ 事务的性能测试问题。

我们将在执行性能测试章节中解决这些难点。

当然，以上 3 点只是一般网站可能遇到的问题。对于不同类型 Web 应用进行性能测试，要在通用的 Web 性能测试指标基础上，针对当前业务流程特点，进一步完善而成。

8.2.11　业务性能分析文档

在整个网站的主要业务流程都被考虑进来之后，可以创建一个业务活动分析文档，以

备之后各相关人员确立性能测试的目标、利用 LoadRunner 等性能测试软件建立用户场景时所使用。业务性能分析文档一般由技术部门代表、需求分析人员等共同编写，描述各个业务流程的软硬件依赖、预计用户数量、业务发生数量、优先等级等。表 8-1 列出了小白所在公司在性能测试之前的部分业务性能分析文档。

表 8-1　XXXX网站业务性能分析文档

业务功能模块名称	软硬件依赖	每天预计用户数量	每天业务发生数量	优先级
用户注册与登录	数据库、网站程序	500	1500	1
用户查询商品生成列表	数据库、网站程序	500	3000	1
用户生成订单	数据库、网站程序、支付接口	100	150	1
用户发帖评价商品、咨询客户服务	数据库、网站程序	500	2000	2
网站人员更新商品	数据库、后台管理程序	50	500	2
网站人员更新订单状态	数据库、后台管理程序	50	500	1

表 8-1 列出的仅仅是一部分，在实际情况中可能会更加细致，比如对于网站人员更新订单状态部分，可以分为出库管理、配送管理等多个操作，可以把每一种操作都记录在表 8-1 这样的文档中。

【每天预计用户数量与每天业务发生数量】

每天预计用户数量简单的理解就是指在一天内，有多少用户在使用这些功能。而每天业务发生数量则是指这些用户在网站上该功能模块中所做的全部操作数量。比如用户注册与登录部分：每天预计用户数量为 500，即有 500 个用户注册或者登录到系统。他们可能是在注册新用户、可能是老用户登录、也可能是老用户在修改个人资料等，因此一般来说，每天业务发生数量要大于等于每天预计用户数量。

有了类似业务性能分析这样的文档，性能测试工程师就可以根据其中的数据来进行 LoadRunner 等工具中的场景设计，从而有针对性地开展测试。

既然提到了指标问题，我们就需要针对当前 Web 应用的性能测试设定一些具体指标，只有它们都达到了，才能说是通过了性能测试。这将是 8.3 节讲述的内容。

8.3　性能测试标准的确定

熟悉 Web 应用的业务流程，有助于我们发现性能测试中的侧重点。俗话说“好钢要用在刀刃上”，与其花费很多时间测试与优化用户很少使用的功能，不如在有限的时间和资源内抓住重点，这样能提高整个过程的效率。

熟悉 Web 应用的业务流程也有利于确定一个合理的性能测试标准。有了测试标准，才能有“合格”与“不合格”的说法。本节将讲解常见的 Web 应用性能测试标准。

【性能测试标准是变化的】

由于网站投入使用后是一个长期的过程，在这期间会发生很多变化：在 Web 应用发布

前确立的某些业务流程可能由于表现一贯稳定，并不需要投入太多的资源；而另外一些业务流程由于市场、用户群等诸多因素的变化而变化。因此，针对以上这些变化，Web 应用的性能测试标准也可能需要一些必要的调整和修订。

8.3.1　确定性能测试目标

要建立适用于实际情况的 Web 应用测试标准，首先要确定性能测试目标。我们在第 4 章曾经介绍过 Web 性能测试的 3 个目标，分别如下。

- 发现系统的代码缺陷：即发现性能相关的 Bug。
- 发现系统的工作能力：可以通过压力测试等方式获得。
- 发现系统性能优化的关键点。

在每一次执行性能测试之前，都要明确当前性能测试的目标是哪一种或者哪几种，从而更具有针对性。不同的性能测试目标所关注的性能测试指标是不完全相同的。如果把所有的性能指标都记录下来，有如下两个缺点：

- 会占用更多的测试时间。记录更多的性能指标会导致整个记录过程略长，但更重要的使分析结果变得更复杂，容易被无关的指标所迷惑。
- 会影响服务器上的 Web 应用。记录全部的性能指标，肯定会伴随较大量的查询、文件写入等操作，会对服务器上的被测试 Web 应用有所影响，也会增加与真实情况下的误差。

因此，测试工程师有必要为每次性能测试制定不同的性能测试目标，采纳不完全相等的指标参数，并且只关注需要被关注的部分。

8.3.2　确定性能测试标准

有了测试目标，下面就可以确定性能测试的标准。确定性能测试标准需要技术部门和需求分析、网站策划人员的参与，最终达成一致，并在测试计划中记录下来。测试标准需要测试指标来证明。

【测试标准与指标的不同】

测试标准是规划测试时确定的总体目的，它考虑了多种因素，相对笼统与抽象。而测试指标则是具体的、可测量的、可以在规定时间内达到的一些度量值，具体直观。比如表 8-2 列出了测试标准和测试指标的例子。

表 8-2　测试目标与测试指标的举例

测 试 标 准	测 试 指 标
系统能够持续稳定地运行	持续性指标：系统运行 5 天时间内，Web 应用无服务中断。 稳定性指标：系统运行 3 天后，响应时间无明显变化，内存使用无显著变化
各个页面能够快速打开以便浏览	首页的响应时间在 10 秒之内 第一页查询结果在 15 秒内能够显示完毕

8.3.3　常见的 Web 应用性能测试指标

常见的 Web 应用性能测试指标主要有如下几种：

- 并发用户数：每天用户数量×（每用户平均使用系统的时间/每天的总时间）。举例来说，假设小白公司的网站每天预计用户数量是 500，预计每个用户在网站中的操作平均时间为一个小时（包括登录、修改个人信息、发帖评论、查询浏览商品信息），则预计的并发用户数为 500×（1/24）=21。
- 吞吐量：可以通过并发用户数量×（每天业务发生数量/每天用户数量）/每用户平均操作时间（以秒为单位）来推算，结果数字的单位可以近似认为是页面浏览数/秒。每天业务发生数量与每天用户数量的比值实际是每个用户平均操作的业务数量，将其与并发用户数量相乘则是并发的这些用户总共的业务数量。最终，将所得除以每个用户平均操作时间，得到每秒钟 Web 应用需要支持的业务数量，即吞吐量。
- 响应时间：一般以响应时间小于 20 秒、10 秒、5 秒为标准。具体数值可以根据不同 Web 应用的实际情况和要求来调整，有的时候在同一个 Web 应用内部，也需要分开设置目标。比如对于网站的首页，由于是所有内容的入口，响应时间较长会产生较大影响，因此要尽可能地短。而对于搜索页面，用户的预期能够容忍稍长一些的响应时间。
- CPU 占用率等硬件指标：这些指标在前面的章节中详细地提到，可以根据需要增加到性能测试指标中来。

【实战演练：一个计算吞吐量的例子】

以并发用户数中所举范例来说，并发用户数为 21，每用户平均操作时间为 3600 秒，每天业务发生数量为 5000，则每秒 Web 应用的吞吐量为：21×5000/3600=29.17 页面浏览数/秒。

8.3.4　性能测试标准范例

表 8-3 列出了小白的公司网站所采用的部分性能测试标准。

表 8-3　XXXX公司网站性能测试标准

测 试 标 准	客户端性能指标	服务器端性能指标
系统能够持续不间断地运行	在测试时间段内（目前暂定为 7 天），Web 应用均可正常访问	服务器内各相关应用程序无 Crash 现象。 网络设备运行正常，网络吞吐率不出现中断
系统能够稳定运行	在测试时间段内，测试用户稳定的情况下，从客户端访问 Web 应用的响应时间不应显著变化	服务器内各主要性能计数器数值稳定，平均值不超过 70%
系统能够承受压力运行	在高负荷的情况下，Web 应用依然能够给客户端以反馈	在高于预计用户数量 1 倍的情况下，服务器 CPU 与内存平均使用不高于 90%

续表

测 试 标 准	客户端性能指标	服务器端性能指标
系统能够经受峰值运行	在集中负荷的情况下，Web 应用依然能够给客户端以反馈	向数据库集中插入大规模数据、同时上传较大文档时，服务器内各相关应用程序运行正常

8.4　编写性能测试计划

在部门内部甚至更大的范围内开展讨论，直至确立性能测试标准之后，性能测试工程师就可以结合自身的工作情况，着手编写性能测试计划。性能测试计划应该包括如下几部分。

（1）性能测试计划本身的信息：性能测试计划的编写人、实施人、负责人、版本、编写日期、审核日期、批准日期、编写目标等。

（2）被测试 Web 应用的信息：包括项目或网站的介绍、物理或逻辑结构图、简单的业务流程、目标用户、开发测试所需平台与技术、相关技术文档的索引与位置等。

（3）性能测试的时间、工作安排简述：性能测试参与的人员、使用的工具、测试结果的保存；性能测试开始时间、结束时间等。可能的话，要将工作的几个阶段列出，比如在某一时间点之前，完成性能测试环境的配置；在某一时间点之前，完成性能测试的执行等。这一部分的内容是提供给非技术部门的相关人员阅读的，便于他们了解性能测试的总体安排，从而能和其他部门的工作配合起来。

（4）具体的性能测试工作安排：具体到一定时间（一般以天为单位）的工作进度安排，包括图 8-1 中所有的阶段细分。这项信息将提供给技术部门的相关人员阅读。

8.4.1　性能测试人员组成

从本章即可以看出，性能测试不单单是运行 LoadRunner 等测试工具那样简单，还需要很多人的配合。因此，测试计划要把他们都列出来。性能测试团队一般包括测试环境配置工程师、性能测试工程师、测试经理以及其他相关人员诸如网络管理、开发工程师等。

在测试计划中还要考虑团队中工作的依赖关系，这种关系也将反应到 8.4.2 节关于时间的安排上来。举例来说，性能测试需要模拟真实的生产环境，因此需要测试环境配置工程师搭建、配置测试环境，在性能测试工程师、网络管理和测试经理认可后才能开展实际的性能测试。

8.4.2　性能测试工具的选择

在测试文档中，一般要解释技术上选择性能测试工具的原因。这个原因往往是该性能测试工具相对于其他工具的特长之处，同时也可能体现 Web 应用所选择技术平台的特点。列出以上这些信息是很有好处的：它有利于今后工作经验的总结和公司内部知识的积累。

8.4.3　性能测试进度安排

性能测试计划要重点列出时间安排和人员安排，如表 8-4 所示。

表 8-4　XXXX公司网站性能测试进度计划

阶 段 名 称	起 止 时 间	阶段交付成果	参 与 人 员
测试准备（计划、目标和测试标准确定）	2008.10.8～2008.10.12	测试计划文档	测试经理、小白、需求分析部、开发工程师等
测试环境设计	2008.10.12～2008.10.14	测试环境文档	配置工程师小黑
测试用例、工具脚本开发与调试	2008.10.12～2008.10.20	测试用例 LoadRunner 脚本	小白
测试工具部署与测试场景创建	2008.10.15～2008.10.16	测试环境 测试部署文档	小黑、小白
执行性能测试	2008.10.17～2008.10.24	测试结果记录	小白
执行压力测试	2008.10.25～2008.10.26	测试结果记录	小白
执行负载测试	2008.10.27～2008.10.28	测试结果记录	小白
测试结果分析、生成测试报告	2008.10.29～2008.10.30	测试分析报告	测试经理、小白
性能测试总结、优化讨论	2008.11.1～2008.11.2	性能测试总结、性能优化讨论会议记录	测试经理、小白、开发工程师等

这样的表格在测试计划中是必须的，有必要的话，可以用前面章节介绍过的 Project 项目管理软件进行工作的细分。对于人员安排，Project 也有很好的分配管理方式。

【实战演练：使用 Project 分配资源】

如图 8-9 所示，在 Project 中为项目分配人员的方法：在为项目添加人员之后，可以通过双击每个人员的记录来设定他们的有效工作时间等信息，如图 8-10 所示。

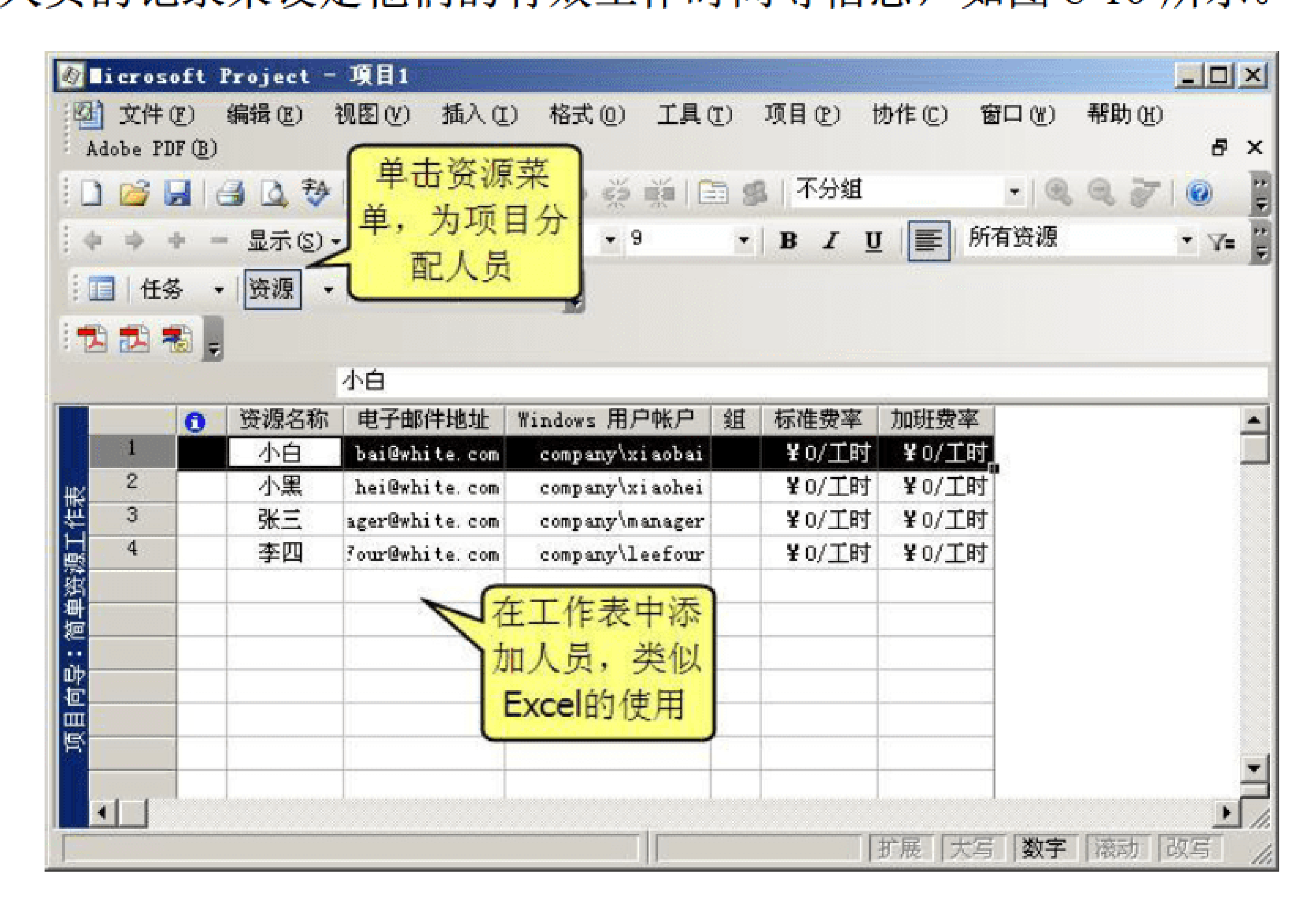


图 8-9　在 Project 中为项目增加和分配人员

对于更加详细的 Project 使用说明，请感兴趣的读者自行查阅相关文档。

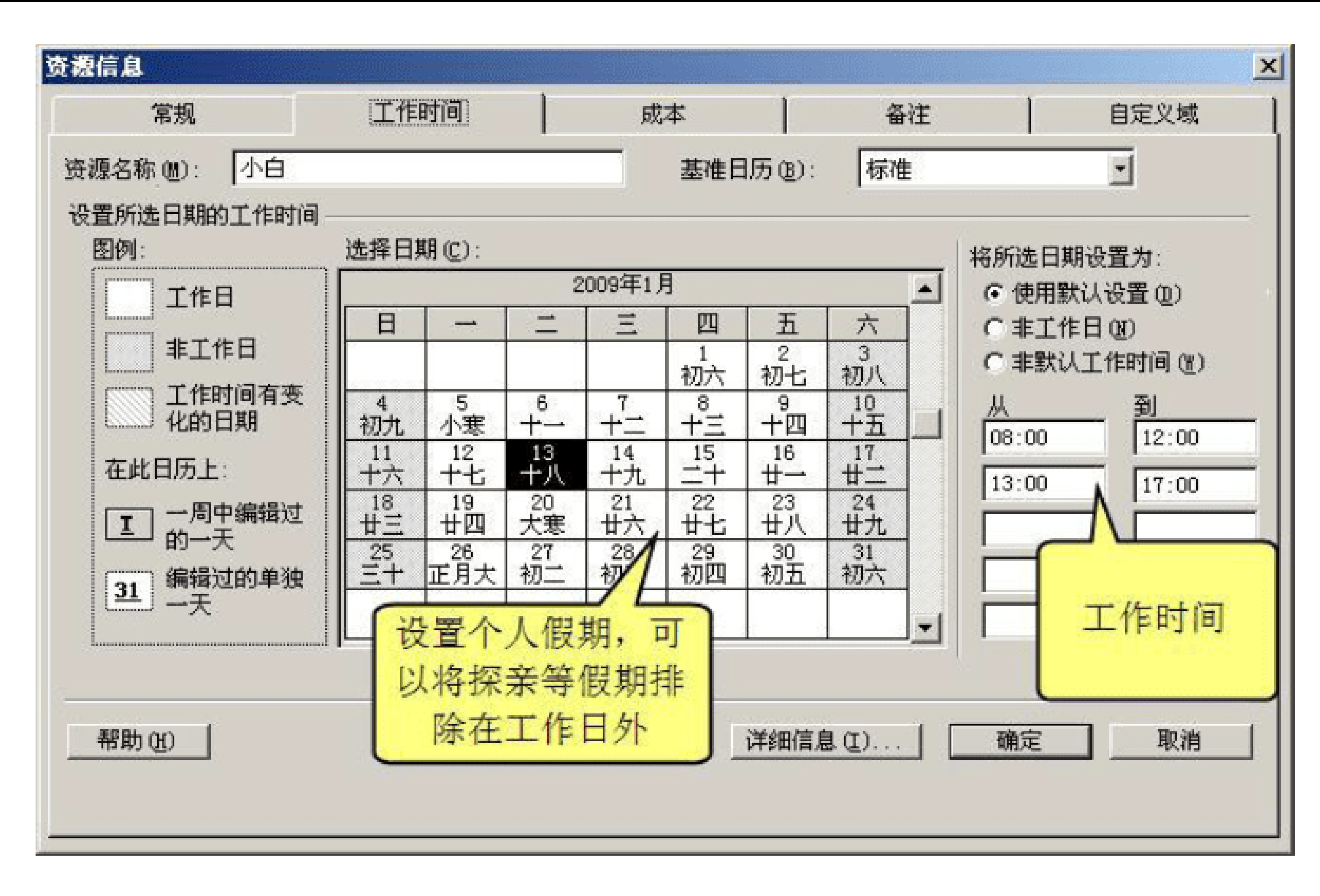


图 8-10　为项目中人员设置可用工作时间

8.4.4　性能测试计划模板

在本章的最后，表 8-5 对小白所编写的性能测试计划目录做了解释。对于目录中某些部分的详细内容，在前文已经列出。读者可以根据实际工作情况，编写更符合实际的性能测试计划。

表 8-5　XXXX公司网站性能测试计划目录

目　录	解　释
第一章　性能测试计划文档信息	包含性能测试计划的编写人、实施人、负责人、版本、编写日期、审核日期、批准日期、编写目标等，每次修改要相应更改或增加记录
第二章　XXXX 网站简介	
第一节　XXXX 网站的背景与结构	简介网站的背景、最终用户的大致构成、网站所采用的开发平台与主要技术、网站系统结构图
第二节　XXXX 网站业务性能分析	列出主要业务流程、预计用户数、吞吐量等
第三章　XXXX 网站性能测试方案	
第一节　本计划中使用的性能测试术语	对使用的术语进行定义，统一认识
第二节　性能测试目标与标准	列出确定后的性能测试目标、标准以及相关指标
第三节　性能测试方法	列出进行的性能测试种类
第四节　性能测试工具选择	列出选择当前性能测试工具的理由
第四章　XXXX 网站性能测试环境与团队组成	
第一节　性能测试环境物理结构图	列出性能测试环境的物理结构图
第二节　性能测试环境软硬件列表	
第三节　性能测试团队组成人员	列出各角色负责人员
第五章　XXXX 网站性能测试计划进度	
第六章　XXXX 网站性能测试风险分析	列出在性能测试中可能存在风险的部分以及解决方案： 能否按时完成？ 结果如何保证准确？ 等

续表

目　　录	解　　释
第七章　XXXX 网站性能测试结果记录	列出测试结果的存放位置、测试报告需要包含的信息
第八章　附录	列出性能测试相关的开发、测试文档位置、链接等信息

8.5　本章小结

本章首先介绍了 Web 应用的物理、逻辑和系统结构，以及典型网站的业务流程。基于这两部分的较好理解有利于发现和确认性能测试目标。实际上，对于每一次性能测试来说，由于处于项目不同的阶段、要满足不同的需求，性能测试目标都有可能发生变化。

性能测试通过与否是通过验证测试结果是否符合性能测试标准来实现的。性能测试标准需要由性能指标（响应时间、吞吐量、服务器的各种性能计数器）等具体数值来量化。在一个好的性能测试计划中，要包含以上内容。

本章最后介绍了编写性能测试计划的要点。俗话说，良好的开始是成功的一半。有了明确细致的计划，并在实际工作中很好地坚持，才能为下一步打好基础。

在第 9 章里，我们将学习配置性能测试环境的一些要点和技巧。

第 9 章　配置测试环境

测试计划编写完毕之后，按照第 8 章开始介绍的流程图，小白即开始着手考虑测试过程与测试环境的问题。测试过程实际就是测试用例、测试脚本的实现过程，可以利用前面几章介绍的 LoadRunner 脚本录制、修改技术来实现。一般来说，在测试用例或者测试脚本准备得差不多的阶段，要在测试部门内召开一次或者多次会议，大家对已经完成的测试用例进行讨论，主要是针对如下两点发现问题。

（1）现有的测试用例是否较大程度地覆盖了被测试产品的各个方面？具体到性能测试来说，就是测试计划中列出的性能测试场景，是否都有对应的测试用例来验证？

（2）测试用例或者测试脚本的实现方法是否合理，是否会带来风险？比如，实现方法本身就是有错误的，因此使用它测试出来的结果自然对整个质量改进意义不大。俗话说“三个臭皮匠，顶一个诸葛亮”。通过测试工程师给大家介绍自己的方法，部门同事是有机会发现个人考虑不周之处的。

在测试过程大致固定并接近设计完毕的同时，就需要准备测试环境。在实际工作场合，公司的测试部门内，有关测试环境的建立与维护往往需要性能测试工程师与测试配置管理工程师或者网络管理员协同完成。

所谓测试配置管理，是软件配置管理的一种，后者简称为 SCM（Software Configuration Management）。通俗地说，软件配置管理者的主要职责在于管理软件的版本，并管理软件测试相关的环境。这方面的知识并不属于本书的内容，感兴趣的读者可以参阅相关专业书籍。

9.1　测 试 环 境

所谓测试环境，就是指测试工程师在进行测试过程时所处的软件、硬件环境。我们在网站上经常能看到电子产品、汽车产品的评测，为了保证结果的客观性，测试者都会在测试报告的某个位置列出测试是在什么条件下进行的。比如图 9-1 所示的就是某权威机构针对汽车进行某种碰撞测试时的测试条件。

与以上这些测试类似，在软件开发行业，测试环境也是同样重要的。对于软件中的 Bug 来说，有些是需要在特定的环境下才会出现的，因此，研究、解决、验证这些 Bug 都需要复现这样的环境。在本节我们将介绍测试环境的分类和准备原则。

9.1.1　准备测试环境的益处

在熟悉了 Web 应用的系统结构和业务流程之后，可能的情况下，我们要开始准备一套

相对简单的测试环境来模拟 Web 应用投入使用后的用户层、表现层乃至数据层。这样做的好处有：



图 9-1　针对汽车进行某类碰撞测试时所采取的测试条件

（1）一般来讲，测试环境可以使技术部门在真正部署 Web 应用之前就获得更多的测试、管理、配置、优化经验。通过模拟一套生产环境，可以开展各种性能、安全、兼容性测试等，较准确地考察软件、硬件对于 Web 应用性能的影响，为今后真正地系统优化做准备。

（2）特别地，对于用户层而言，可以尽可能覆盖更多终端用户的真实条件。一旦 Web 应用投入使用，终端用户可能使用各种操作系统和浏览器的组合访问它：比如 Windows XP、Windows 2000、各种各样的 Linux、Safari 浏览器等，而网站技术部门的电脑为了维护方便，一般都是同一种系统，无法获知 Web 应用在不同组合下的表现。我们建立的测试环境中应该尽可能多地拥有这些组合，以保证浏览正常。

因此，在专业网站的技术部门，可能要维护 3 套 Web 应用环境，如图 9-2 所示。这 3 套环境的特点分别如下。

- 开发环境：由技术部的开发人员建立和使用。在这个环境中，表现层的软件代码随时会变更（由于增加、实现新功能、修改 Bug 等原因），数据层一般均为测试数据。开发环境位于技术部内部。
- 测试环境：由技术部的测试人员在某个稳定的 Web 应用代码基础上建立。在每隔一定周期，比如每天、每周，或者是目前开发的代码通过一定质量标准的情况下创建或者更新。一般处于技术部门内部的服务器中，也可以处于互联网信息中心的服务器机房。
- 生产环境：Web 应用的最终上线环境，绝大多数处于互联网信息中心的服务器机房。

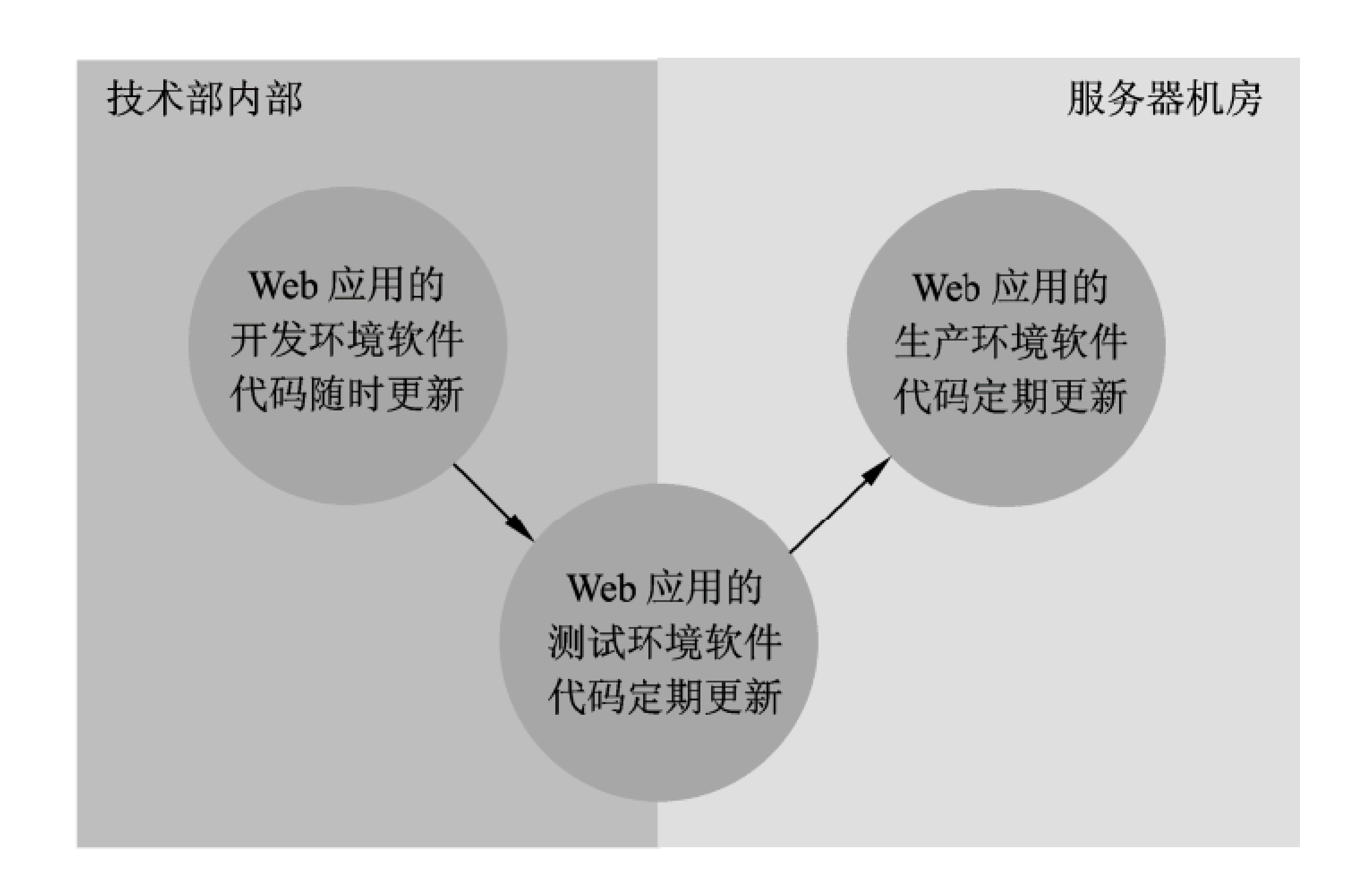


图 9-2　3 种环境：开发环境、测试环境和生产环境的关系示意图

通常情况下，都是现有开发环境，开发进展到一定程度后才有测试环境，测试稳定后上线，最终形成生产环境。

9.1.2　准备测试环境的原则

从前面介绍的内容中可以发现，测试环境对于测试结果可信度的重要性。一个不符合规范、不符合实际情况的测试环境，其上运行测试得出的数据并无多大的参考意义。因此，针对测试环境的准备，有必要总结出几个要点，供测试工程师参考。准备测试环境的原则如下：

- 尽可能地模拟真实生产环境。软件最终是要应用在生产环境当中的，因此，测试环境如果能较准确地模拟真实环境，将非常有利于提高软件在实际应用中的使用质量。
- 保持一致性。在验证 Bug 阶段，由于 Bug 一般都发生于特定的条件和环境中，要保持验证时的测试环境与记录 Bug 时的测试环境相一致。
- 在满足前两条的同时，可以利用一些技术（比如虚拟化）尽量减少环境配置时间，提高真正用于测试过程与验证 Bug 的时间比例。

对于前两条，我们可以通过按照软件说明书等相关文档小心、细致地准备测试环境来达到。对于第 3 条，取决于实际情况，并有赖于不断的摸索经验。9.2 节将通过具体的工具软件，使用虚拟化技术来提高测试环境准备效率的方法。

9.2　虚拟化在准备测试环境中的应用

在 9.1 节中提到了准备测试环境的优点，其中很重要的一条是模拟用户真实访问条件。在实际工作中，模拟这么多的组合往往有困难。

- 技术部门的预算和机器有限，不可能同时安装那么多的操作系统组合；

- 如果维持现有电脑数量，则需要反复地安装、更换系统，耗费大量时间。

为了解决这样的矛盾，我们可以采用虚拟化技术。

9.2.1　虚拟化技术

所谓虚拟化技术，简单说来，就是将 CPU 等硬件分离开来，使得一台电脑能同时运行多个操作系统的技术。

虚拟化技术总体而言，有硬件虚拟化技术和软件虚拟化技术之分，两者有时候需要互相支持与配合。硬件虚拟化技术指的是 CPU 本身就提供了虚拟化的功能，虚拟出来的操作系统运行性能较好。而软件虚拟化技术则指的是只利用软件达到同一硬件多个操作系统同时运行的目的，性能相比硬件虚拟化技术要差，但由于不需要 CPU 支持，成本有所降低。

在硬件虚拟化技术方面，Intel 公司有采用 Intel VT 技术（全称 Virtualization Technology）的 CPU；而 AMD 公司对应的技术叫做 AMD VT。

【软件与硬件结合的虚拟化技术】

有的软件虚拟化技术需要硬件虚拟化技术的支持，比如微软的 Hyper-V 只能安装在包含支持硬件虚拟化技术 CPU 的电脑上。这是需要特别注意的。

虽然在最近一段时间，虚拟化技术表现得更为流行，各种软件产品也较多，但实际上这个领域是在 1966 年由 IBM 公司开创的，最早应用于它的大型机产品上。因此可以说虚拟化技术并不是一个年轻的技术，具备较长的历史。

【查看电脑是否支持硬件虚拟化技术】

以安装有 Intel 公司 CPU 的电脑为例，在开机的时候进入系统的 BIOS 设置，如果在系统配置信息内出现类似图 9-3 这样的界面（具体显示位置会随 BIOS 厂商、CPU 型号不同而有所区别），即可判断出当前电脑支持硬件虚拟化技术。

在某台电脑中虚拟化出来的新操作系统业内一般称为客户系统（Guest OS），也可直接称为虚拟机。而该电脑则被称为主系统（Host OS）或者宿主。宿主和客户系统之间利用虚拟化软件来联络。它们三者之间的关系如图 9-4 所示。

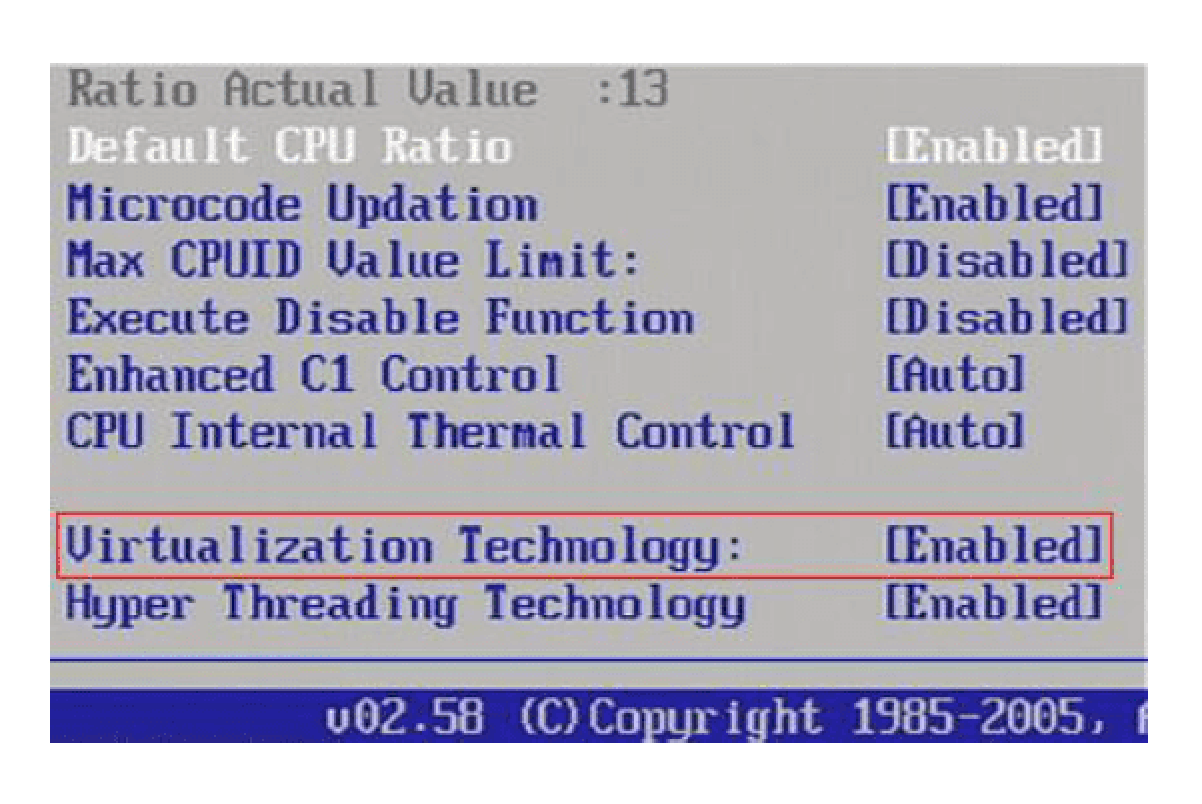


图 9-3　查看当前电脑 CPU 是否支持虚拟化技术

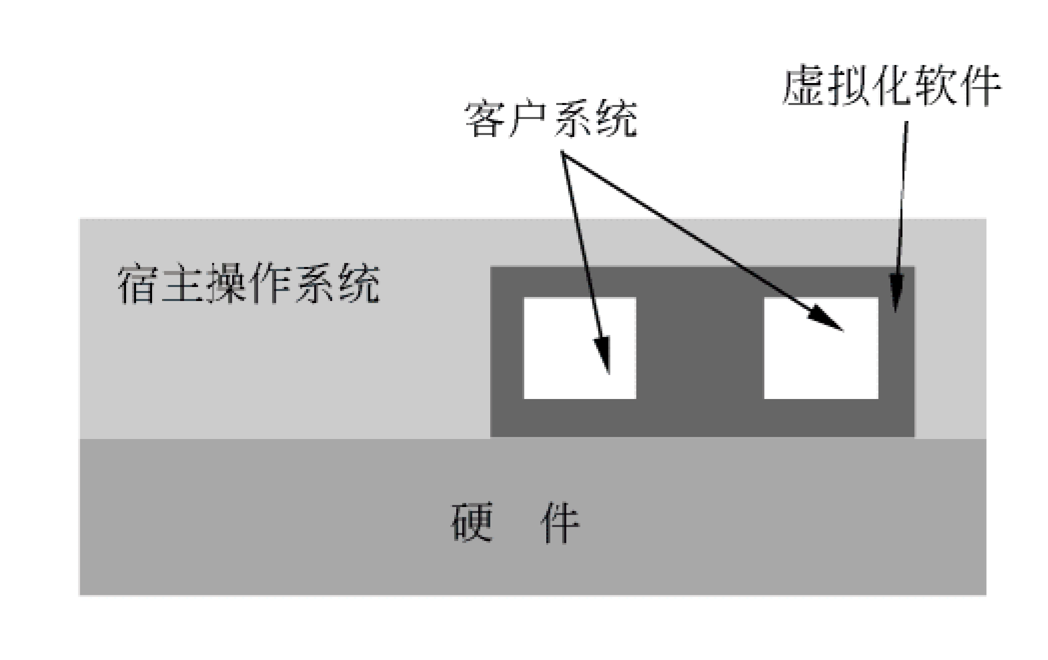


图 9-4　主系统、客户系统、虚拟化软件之间的关系

一个宿主操作系统中可以包含多个客户系统，至于能容纳客户系统的数量是多少，一般取决于宿主的硬件性能和指标。

图 9-4 只是一个简单的表示，实际的虚拟化技术是非常复杂而且在不断变化的。为了

有效地创建、管理这些客户系统，虚拟化软件应运而生。在 9.2.2 节，本书将介绍一些常见的虚拟化软件，并将以 Sun 公司出品的 VirtualBox 为例具体讲述该类型软件的使用。

9.2.2　常见的虚拟化软件

不管是硬件虚拟化技术还是软件虚拟化技术，虚拟化软件都是必备的。虚拟化软件用来创建、管理虚拟机。

各大 IT 公司都有自己的虚拟化解决方案，比如：

（1）微软开发的虚拟化工具系列，它们都只能安装在 Windows 系统之上。其中：

- Virtual PC 用于虚拟化桌面系统，截至 2008 年底最新版本为 Virtual PC 2007 sp1。下载地址如下：
  http://www.microsoft.com/windows/downloads/virtualpc/default.mspx。
- Virtual Server 用于虚拟化服务器系统，截至 2008 年底最新版本为 Virtual Server 2005 R2 sp1。下载地址如下：
  http://technet.microsoft.com/en-us/virtualserver/default.aspx。
- Hyper-V 是新一代的 Virtual Server，目前只能在 Windows2008 64 位系统上采取添加组件（Add Feature）的方式来安装。

（2）VMWare 公司的 VMWare 系列产品，有 VMWare Workstation、Server 等，它们既有 Windows 版本，也有其他操作系统的版本。

（3）开源的虚拟化产品，比如 Sun 公司的 VirtualBox 工具软件，它也可以在多种系统下安装。

在本书中，将在 9.3 节介绍 VirtualBox 的安装、管理以及在准备测试环境中的应用。对于其他的虚拟化产品，各有千秋，但主要功能基本类似，读者可以根据工作中的实际情况进行选择。

9.2.3　虚拟化软件在软件测试中的应用

在软件测试过程中使用虚拟化软件，主要目的为了完成下列任务：

（1）构建与模拟生产环境，利用虚拟化软件大都支持的恢复、还原、快照等功能，快速完成不同配置系统的切换，从而节省部署操作系统、配置生产环境的时间，增加真正用于测试产品的时间，提高工作效率。

（2）在不清楚产品对于某个平台的支持情况时，可以先利用虚拟化软件进行模拟，可以预先发现和了解在该平台下真正执行测试过程中可能遇到的困难和解决办法。

9.3　VirtualBox 实战

本节以使用 VirtualBox 安装一台 Windows XP 虚拟机为例，为在实际工作中利用虚拟化技术完成测试环境的准备打下良好的基础。对于前文列举的其他虚拟化软件，其操作过程、软件界面基本大同小异，相信读者经过短暂的一天熟悉过程，就可以掌握得很好。

9.3.1　VirtualBox 简介与安装

VirtualBox 是 Sun 公司推出的开源虚拟化产品，因此使用者需要访问 Sun 的官方网站（http://www.sun.com）来获取，如图 9-5 所示。读者也直接访问 http://www.virtualbox.org 进行下载，如图 9-6 所示。

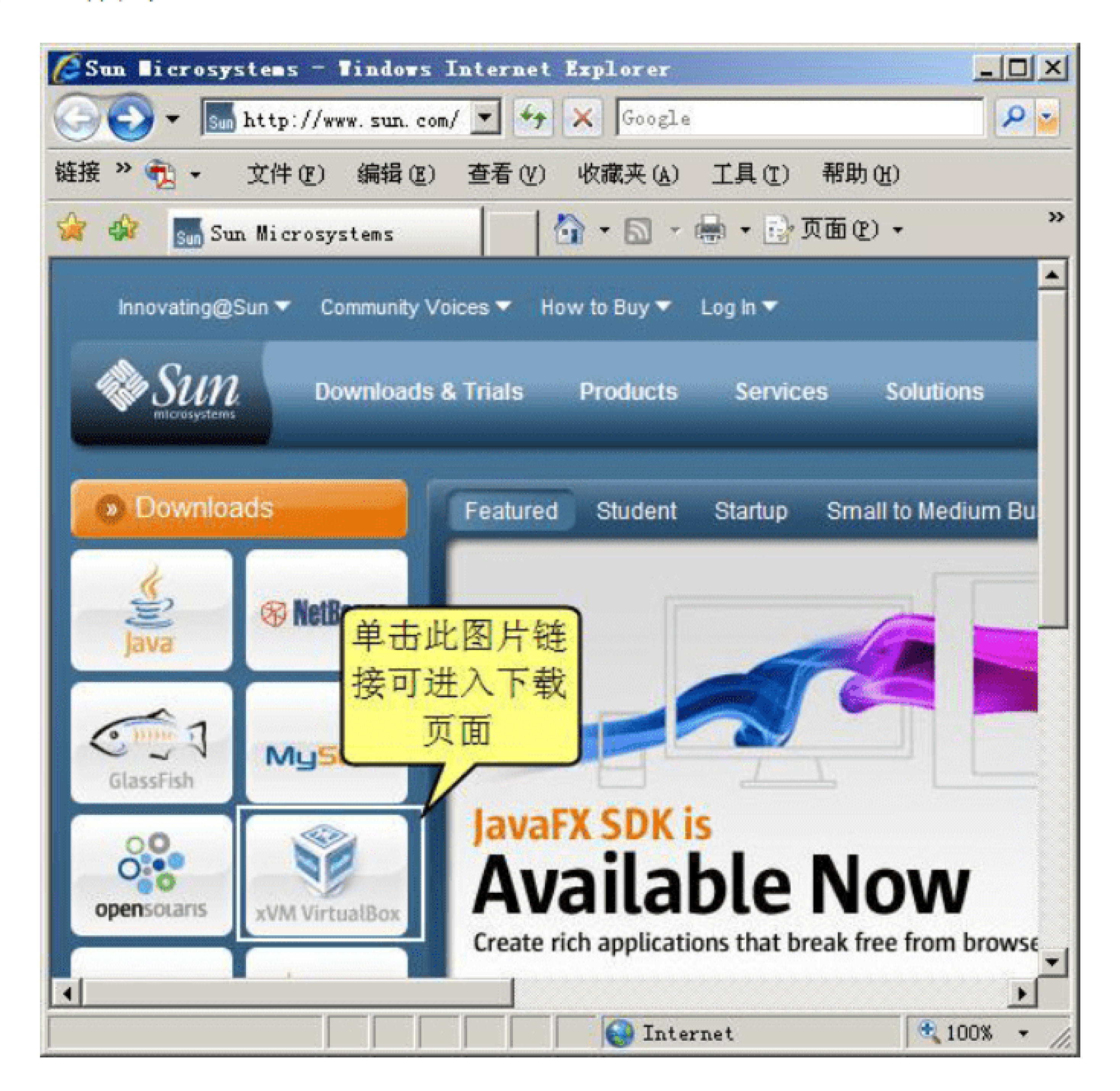


图 9-5　Sun 公司官方网站上的 VirtualBox 链接



图 9-6　VirtualBox 的官方网站也提供了下载链接

在下载页面中，根据读者机器类型的不同（比如 Windows x86 平台还是 Linux 平台等），选择合适的安装包，保存至本地硬盘中。截至 2008 年 12 月，VirtualBox 的最新版本为 2.0.6，

在 Windows x86 系统下的安装文件名为 VirtualBox-2.0.6-39760-Win_x86.msi。

鼠标双击下载后的安装文件，即可开始安装，整个过程非常简单，无须特别设置，只要保持默认设置，连续单击 Next（下一步）按钮即可。安装完毕后，在程序菜单会生成名为 Sun xVM VirtualBox 的程序组。

9.3.2　VirtualBox 管理菜单介绍

单击 Windows 系统任务栏中的“开始”|“程序”| VirtualBox 命令，将打开程序主界面，如图 9-7 所示。从图 9-7 中可以看出，VirtualBox 界面并不复杂。在界面上部有 3 个菜单，分别是管理、控制和帮助。下面介绍一下管理菜单的操作。对于控制菜单，将在 9.3.3 节创建虚拟机中介绍。对于帮助菜单，其使用与其他软件并没有什么大的区别，读者在需要查看功能说明、遇到使用问题的时候可以求助于这个菜单。



图 9-7　VirtualBox 的主界面

（1）VirtualBox 的管理菜单共有 3 个子菜单项，依次为虚拟介质、全局设定和退出，如图 9-8 所示。



图 9-8　VirtualBox 的管理菜单

（2）选择“管理”|“虚拟介质”命令，弹出“虚拟介质管理器”对话框，如图 9-9 所

示。虚拟介质包括 3 种：硬盘（Hard Disks）、虚拟光盘和虚拟软盘。单击导航菜单中的“新建”按钮，将出现创建新的虚拟硬盘向导。在界面上直接单击 Next 按钮，系统询问创建何种类型的虚拟硬盘，如图 9-10 所示。

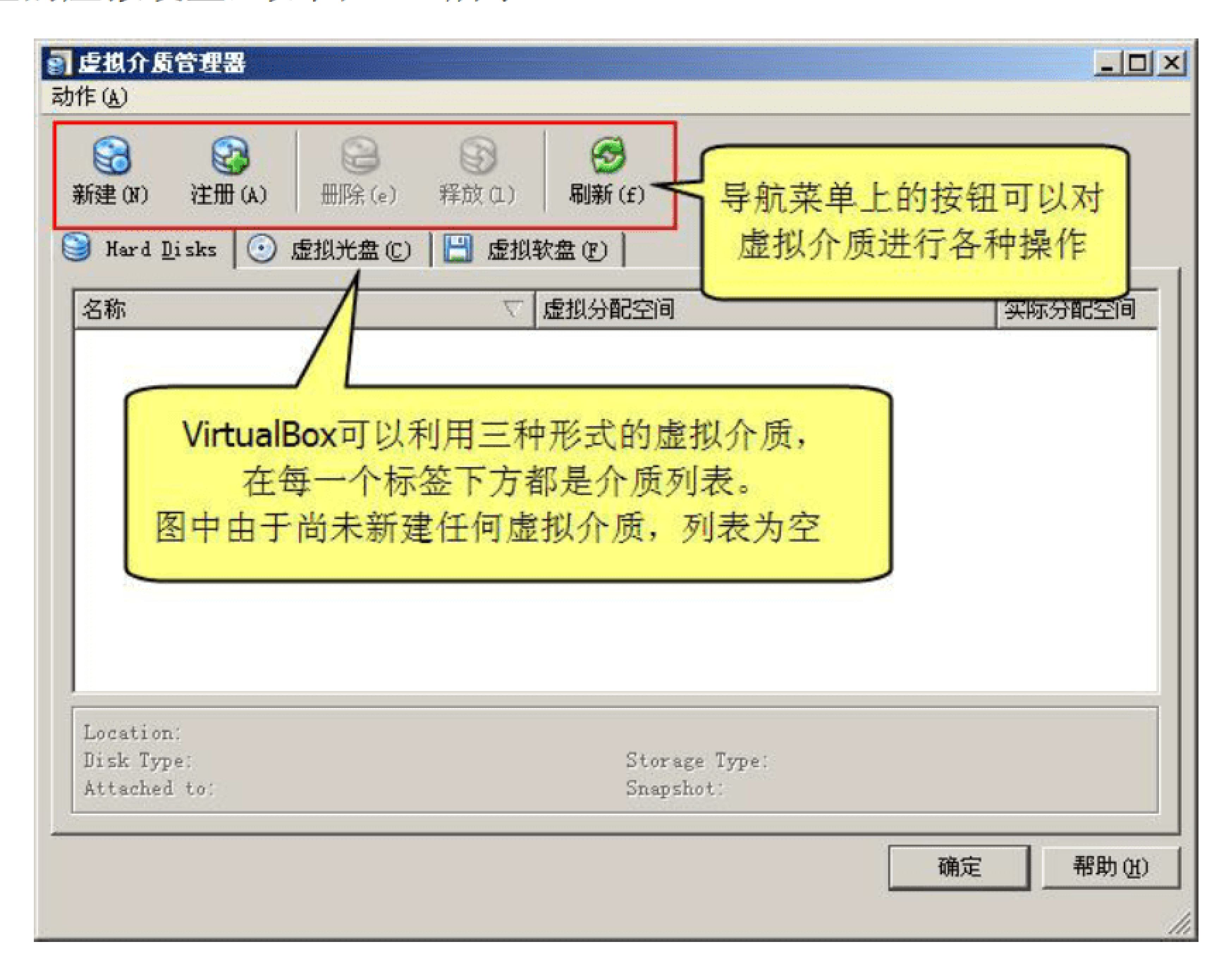


图 9-9　虚拟介质管理器



图 9-10　设定虚拟硬盘类型：动态或者固定

（3）在图 9-10 中，虚拟硬盘有两种，分别是动态扩展映像和固定大小映像。通俗地说，动态扩展映像随着程序的不断使用，其占用实际物理硬盘的空间会不断增长。而固定大小映像则会开始就占用一个固定的空间，之后大小也基本不变。本书在这里维持默认选择：

动态扩展映像，单击 Next 按钮，进入图 9-11 所示界面。



图 9-11　设置虚拟硬盘的大小与映像文件名

（4）在拖动滑块选择好虚拟硬盘大小后，单击 Next 按钮，虚拟硬盘设置完成。虚拟硬盘中存放的数据包括：客户操作系统、系统运行过程中用户创建的各类文件等，可以视作一块虚拟化的物理硬盘。

9.3.3　利用 VirtualBox 设置虚拟电脑配置

设置好虚拟硬盘后，我们就可以开始创建一个新的虚拟电脑。当然，也可以直接创建虚拟电脑，在过程中间同时创建虚拟硬盘。虚拟电脑的操作系统以及其他几乎所有数据将保存在虚拟硬盘中。

（1）在 VirtualBox 主界面的导航菜单中单击“新建”按钮，弹出新建虚拟电脑向导，如图 9-12 所示。单击 Next 按钮。在随后弹出的“虚拟电脑名称和系统类型”对话框的“系

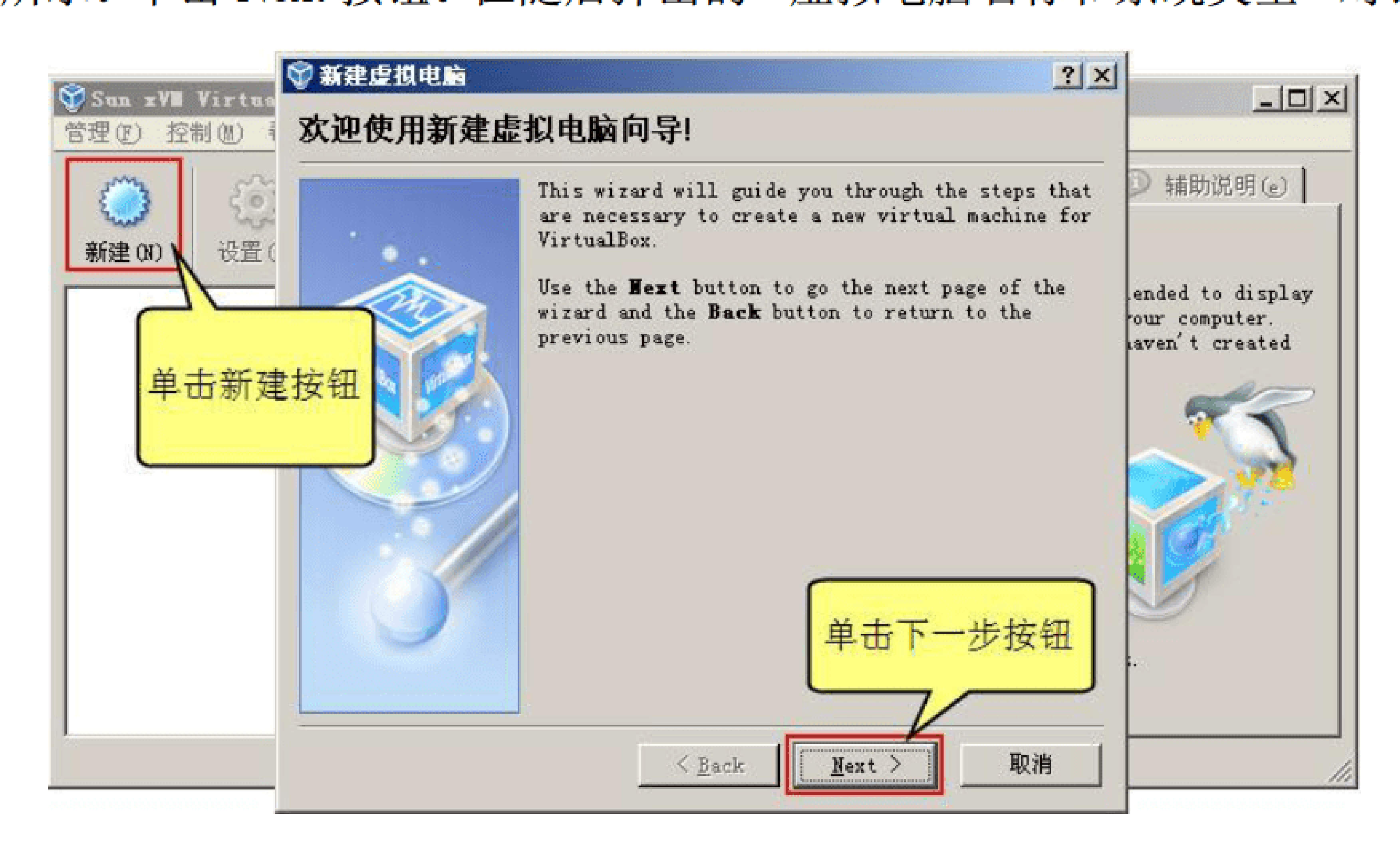


图 9-12　打开新建虚拟电脑向导

统类型”下拉列表框中选择该虚拟电脑的操作系统类型（这里选择的是 Windows XP），并将虚拟电脑名称命名为 WindowsXP，填写在“名称”文本框内。在“系统类型”下拉列表框中我们可以清楚地看到 VirtualBox 支持多种操作系统，包括 Windows 以及列为 Other 类（比如 Solaris 等）的操作系统，如图 9-13 所示。



图 9-13　设置虚拟电脑名称以及类型

（2）在图 9-13 中设置完毕后单击 Next 按钮，进入虚拟电脑内存设置对话框，如图 9-14 所示。



图 9-14　设置虚拟电脑内存大小

（3）通过调整“内存大小”滑块，或者在右边的文本框中直接输入内存数值，以完成对当前虚拟电脑内存的设置。单击 Next 按钮后如图 9-15 所示。在这一步骤中，需要为虚拟电脑选择启动盘。在图 9-15 的“启动盘”下拉列表框中，列出了当前已经设置好的虚拟磁盘（有关设置虚拟磁盘请查阅 9.2 节）。如果之前并未设置虚拟磁盘，或者需要对现有虚拟磁盘进行更改，可以单击“新建”或“现有”按钮，打开虚拟介质管理器，在其中进行添加、删除和更改操作，如图 9-16 所示。



图 9-15　为虚拟电脑选择或新建可启动的虚拟硬盘

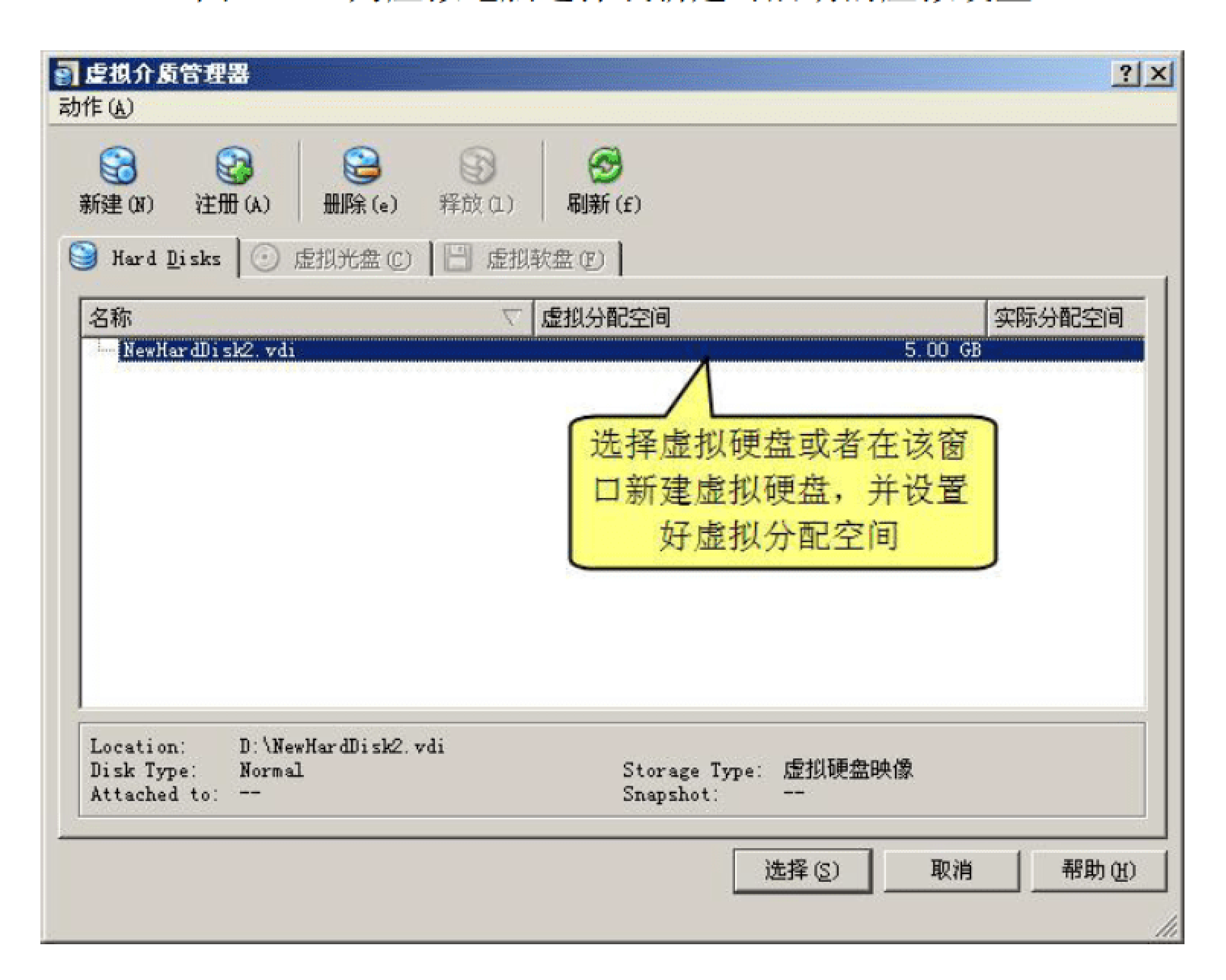


图 9-16　通过虚拟介质管理器选择虚拟硬盘

（4）虚拟硬盘实际就是存在于宿主电脑硬盘上的一个.vdi 文件。它的大小等于图 9-16 中虚拟分配空间的数值。当虚拟硬盘设置完毕之后，整个虚拟电脑的创建接近完成。单击 Next 按钮，VirtualBox 将把正在创建的当前虚拟电脑配置摘要显示在界面上，如图 9-17 所示。如果确认无误，可以单击“完成”按钮关闭新建虚拟电脑向导。这样，一台新的虚拟电脑就已经创建完成了。

此时，我们可以发现，在 VirtualBox 的主界面左侧（即虚拟电脑列表），已经显示出一台名称为 WindowsXP 的虚拟电脑，如图 9-18 所示。当鼠标在该名称上单击选中之后，VirtualBox 主界面右侧将显示选择虚拟电脑的详细配置明细。

不过，截至目前，笔者只是完成了虚拟电脑的创建工作，相当于购买了一台没有预装操作系统的 PC 机，也就是说，虚拟电脑还无法真正投入使用。这一点可以通过下面的小实验来证明。



图 9-17　新建虚拟电脑的配置摘要



图 9-18　单击向导的完成按钮后 VirtualBox 的主界面

（1）在图 9-19 中，即 VirtualBox 的主界面中选择虚拟电脑 WindowsXP 并右击，在弹出的快捷菜单中选择“开始”选项（也可以直接在导航条中单击“开始”按钮），将启动该虚拟电脑，类似物理机器开机的过程。



图 9-19　单击开始菜单或者开始按钮启动选择的虚拟电脑

（2）在启动过程中，无论虚拟电脑中是否安装有操作系统，都会首先弹出一个对话框，如图 9-20 所示。该对话框说明了用于在宿主电脑与虚拟电脑之间切换的组合键，默认为右 Ctrl 键。我们可以通过在 VirtualBox 主界面中的管理菜单中对该组合键进行定义，感兴趣的读者可以自行实验。

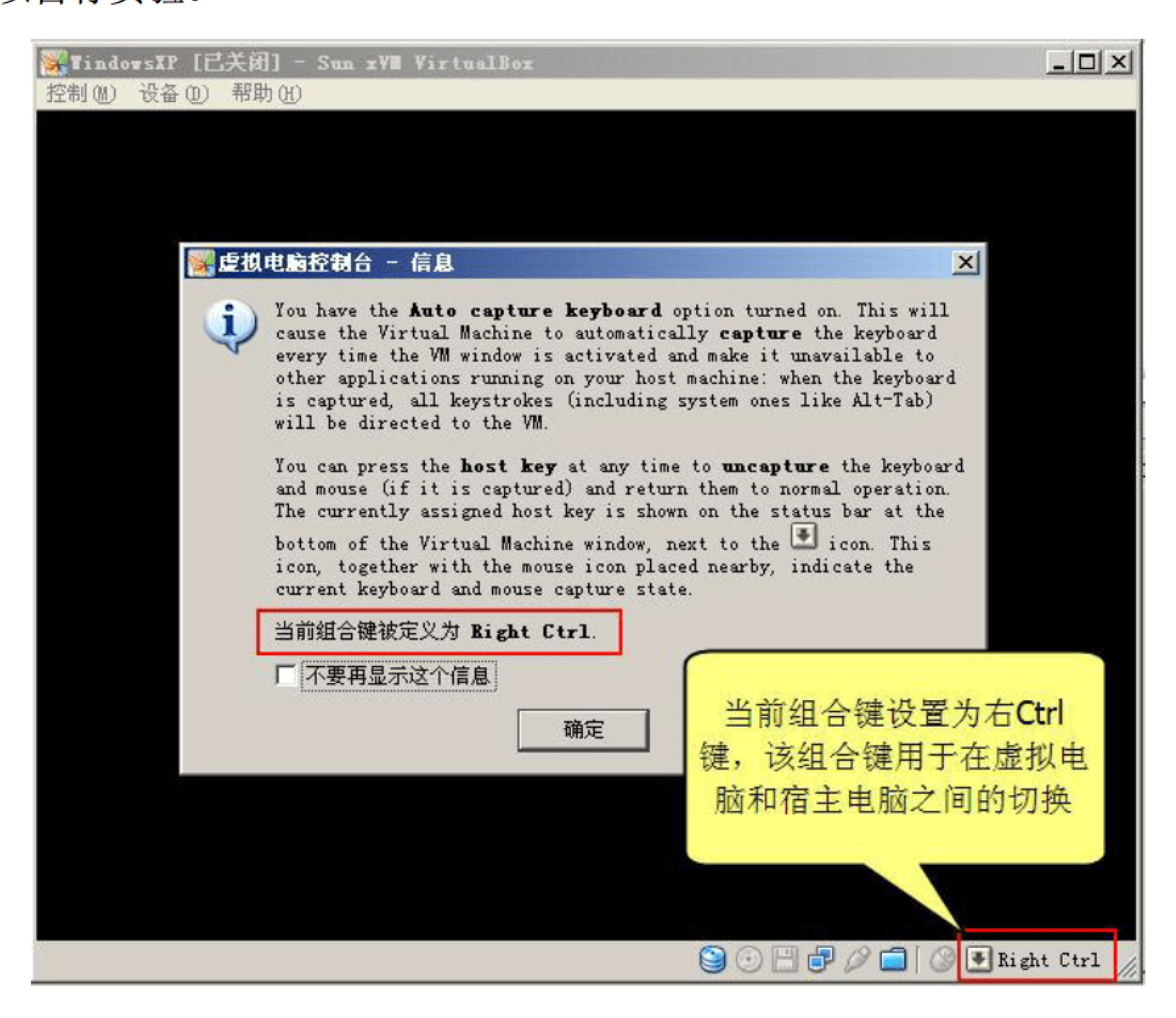


图 9-20　用于在虚拟电脑与宿主电脑之间切换的组合键

（3）但是，截至目前，虚拟电脑内并未安装操作系统，显然是无法启动电脑的，系统会给出类似图 9-21 的错误信息。它提示我们，下一步要做的就是开始 WindowsXP 的安装过程。

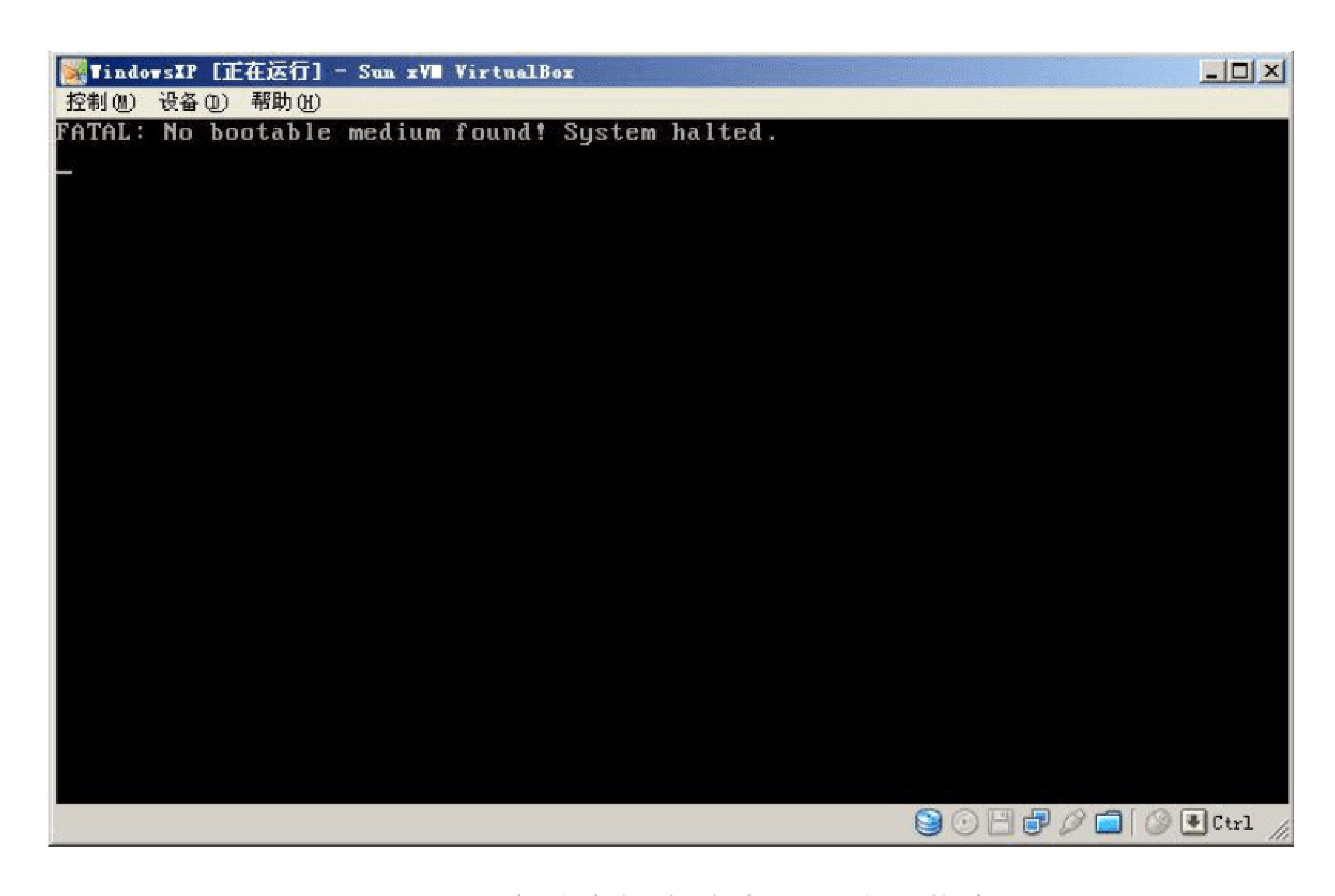


图 9-21　启动虚拟电脑会出现错误信息

（4）为了使得虚拟电脑能够在工作中使用，我们将开始安装操作系统。类似一台新的物理机器的安装，虚拟电脑安装操作系统一般也需要通过光驱来进行。只不过在这种情况下，光驱可以是如下两种：

- 真实的物理设备，即宿主电脑上的真实光驱。将操作系统安装盘放入宿主电脑的光驱中，虚拟电脑即可识别并开始进行安装。
- 虚拟光盘，即保存在宿主电脑某文件夹中的光盘映像文件（以.iso 为文件后缀）。

两种方法可以根据实际情况选用。本书以虚拟光盘为例，简要介绍一下为虚拟电脑安装操作系统的过程。

在 VirtualBox 主界面中选择之前创建完毕的虚拟电脑 WindowsXP 并右击，在弹出的快捷菜单中选择 Start（开始）选项，系统将新弹出一个窗体，用于表示该虚拟电脑的界面，如图 9-22 所示。选择该窗口的设备菜单，选择分配光驱菜单项，单击虚拟光盘，将弹出虚拟介质管理器窗口。

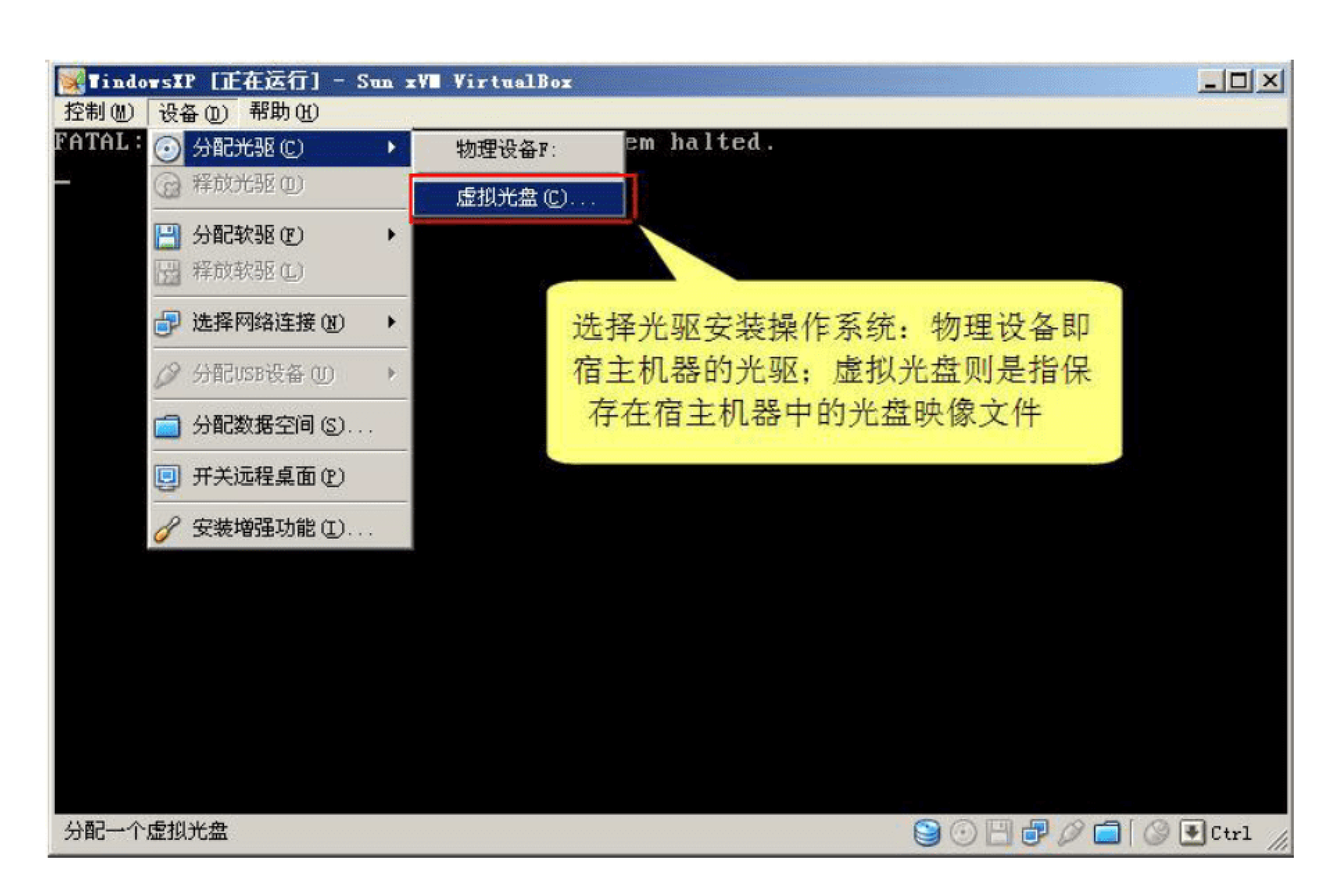


图 9-22　为虚拟电脑分配光驱以便安装操作系统

在虚拟介质管理器窗口的导航条中单击注册按钮，选择宿主电脑上合适的光盘映像文件，如图 9-23 所示。在本书所举出的这个例子当中，光盘映像文件存放在宿主电脑的 D 盘根目录下，文件名为 WINXPSP3CHS.iso，从名称可以猜测到安装文件为中文版本 Windows XP，并且包含 sp3 更新包。

一旦我们通过前面讲述的两种方法之一分配了光驱，在设备菜单下的释放光驱就由灰色变为可用，可以通过单击它来实现更换光盘的功能（比如需要在虚拟电脑中安装若干软件时，光驱是可以装载别的光盘或者 ISO 文件的），如图 9-24 所示。

截至目前，光驱中已经有了 Windows XP 系统的安装光盘（以光盘映像文件形式存在的虚拟光盘），可以开始安装操作系统了。但在正式安装之前，还需要用光盘启动虚拟电脑。方法是依次选择“控制”|“重启”命令即可，如图 9-25 所示。

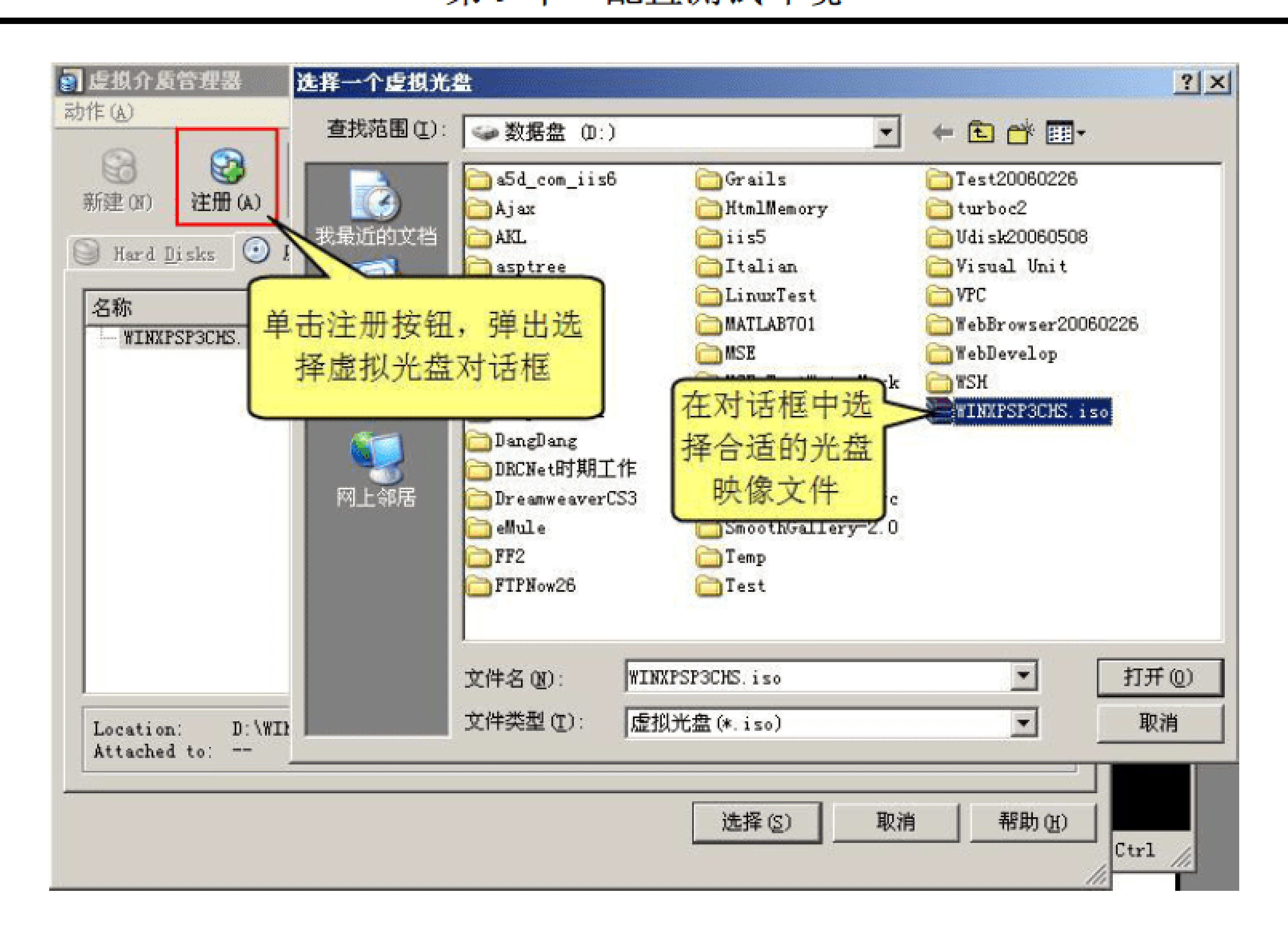


图 9-23　为虚拟电脑选择虚拟光盘即光盘映像文件

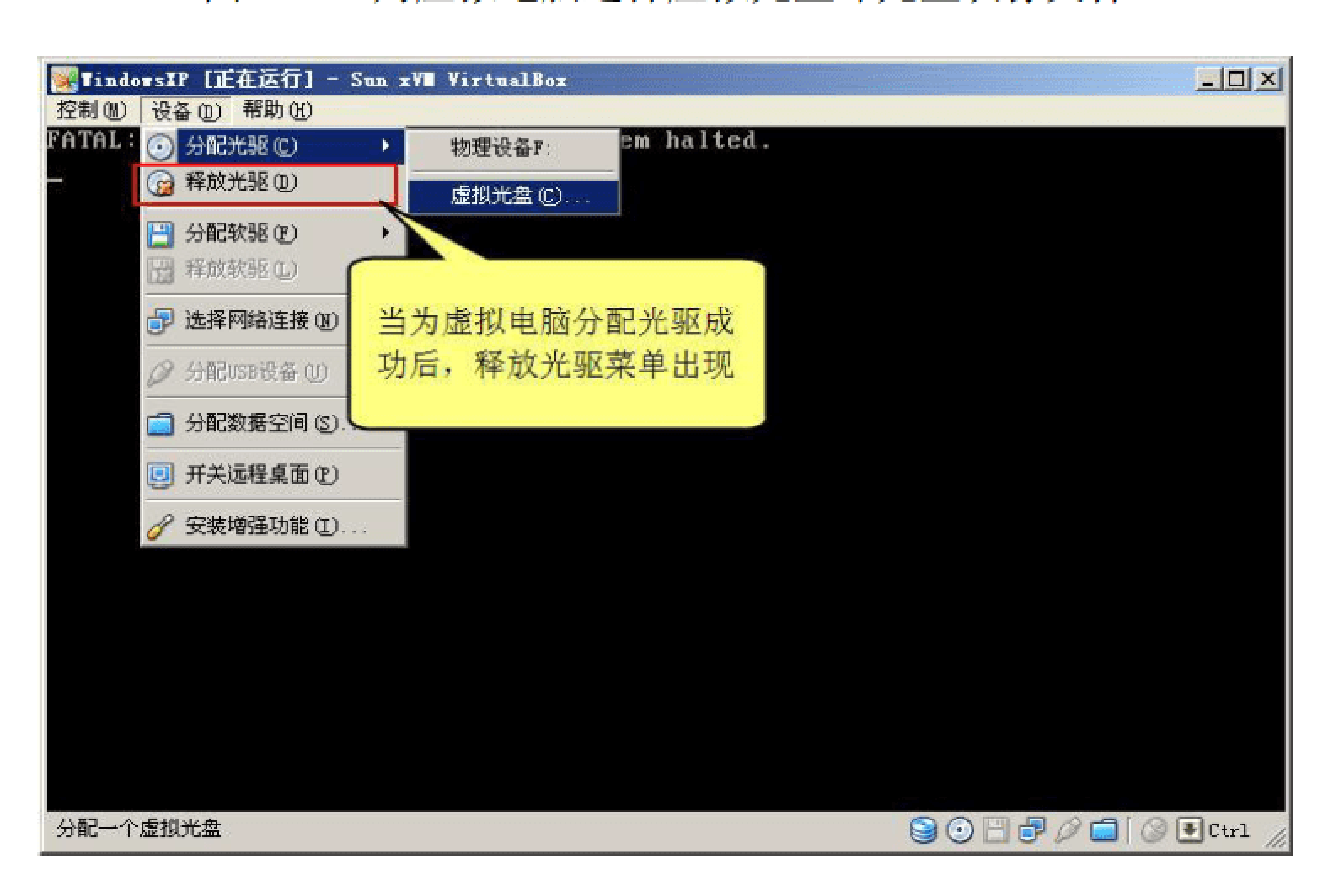


图 9-24　当分配光驱成功后，释放光驱菜单出现



图 9-25　控制虚拟电脑的菜单

当我们选择重启后，虚拟电脑以光驱启动，开始了熟悉的操作系统安装过程。该过程与在真实机器上安装 Windows XP 没有差别，如图 9-26 所示，本文不再详细讲述。

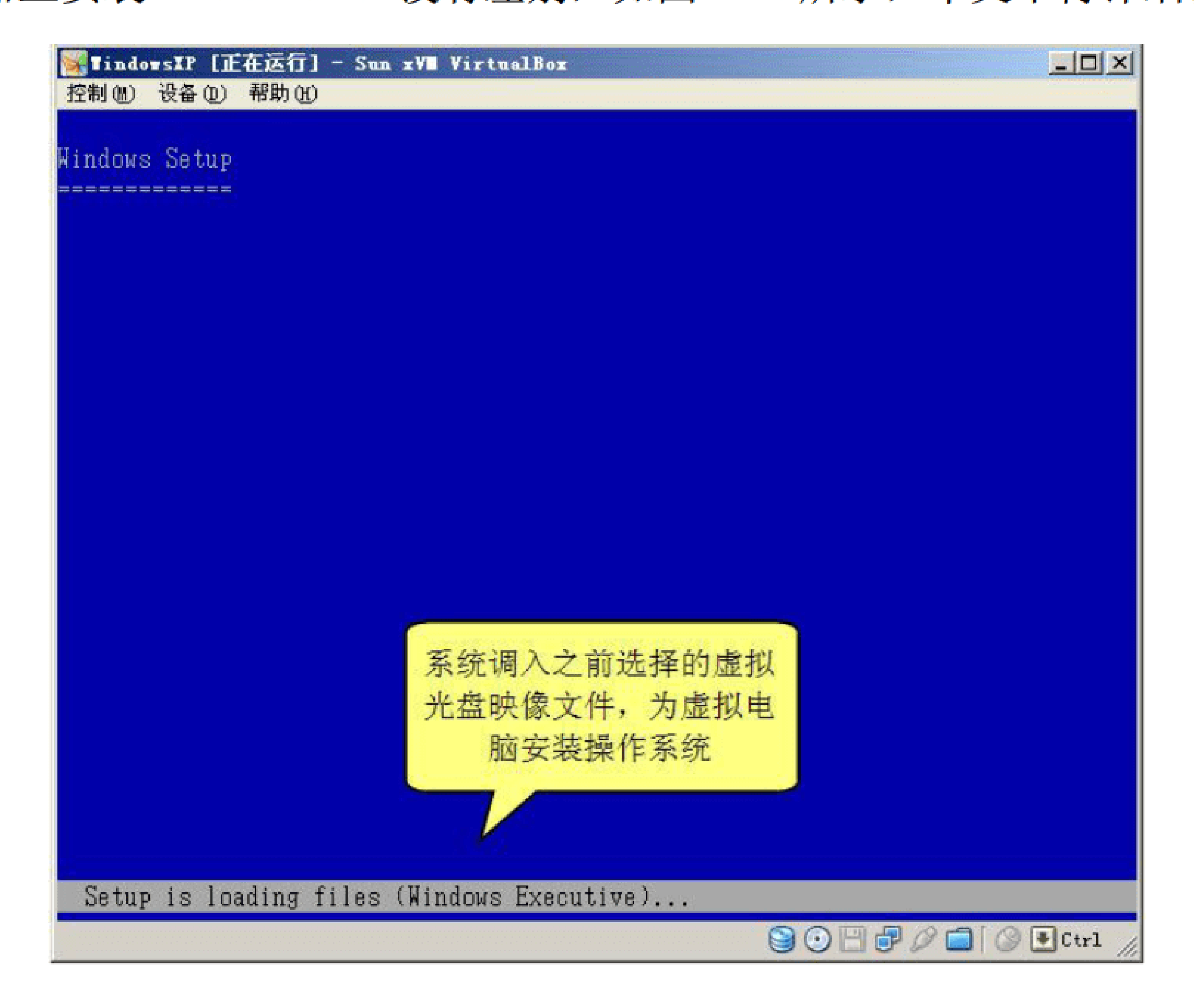


图 9-26　重启虚拟电脑后开始操作系统的安装过程

对于安装其他操作系统，也是采用同样的步骤进行安装。

- 分配光驱，找到操作系统安装文件（以光盘或者光盘映像文件形式存在）。
- 重启虚拟电脑，以光驱启动，开始安装过程。

经过大致 40 分钟的等待，在安装结束后可以发现，虚拟电脑操作系统的界面与初次安装完毕的真实机器完全一样。

9.3.4　安装 VirtualBox 中的增强功能

为了使用上的方便，各种虚拟化软件安装包中一般都包含一个专门的程序，运行在虚拟电脑上，为其提供更好的鼠标、显卡等硬件支持。这样的程序一般被称为 Guest Addin 或者 Guest Additions。VirtualBox 也不例外，但是，在默认情况下，这个程序是不会被自动安装的，需要使用者手工添加。

添加的方法很简单：在打开的虚拟电脑 WindowsXP 控制界面上选择“设备”|“安装增强功能”命令，弹出增强功能安装向导，如图 9-27 所示。该向导不需要修改特别的设置，保持默认，一直单击 Next 按钮即可完成安装。值得一提的是，安装完毕可能需要重新启动虚拟电脑，如图 9-28 所示。实际上，安装增强功能是通过在虚拟电脑的光驱中加载安装光盘来实现的。

9.3.5　与宿主电脑共享文件

为了工作方便，经常需要在虚拟电脑与宿主电脑之间互相复制文件。比如，虚拟电脑

被配置成一台 Web 服务器，每天网站程序员开发的新代码要部署到其上进行一些测试。那么，如何将网站文件复制到虚拟电脑中呢？很容易想到的大致有如下两种方法。

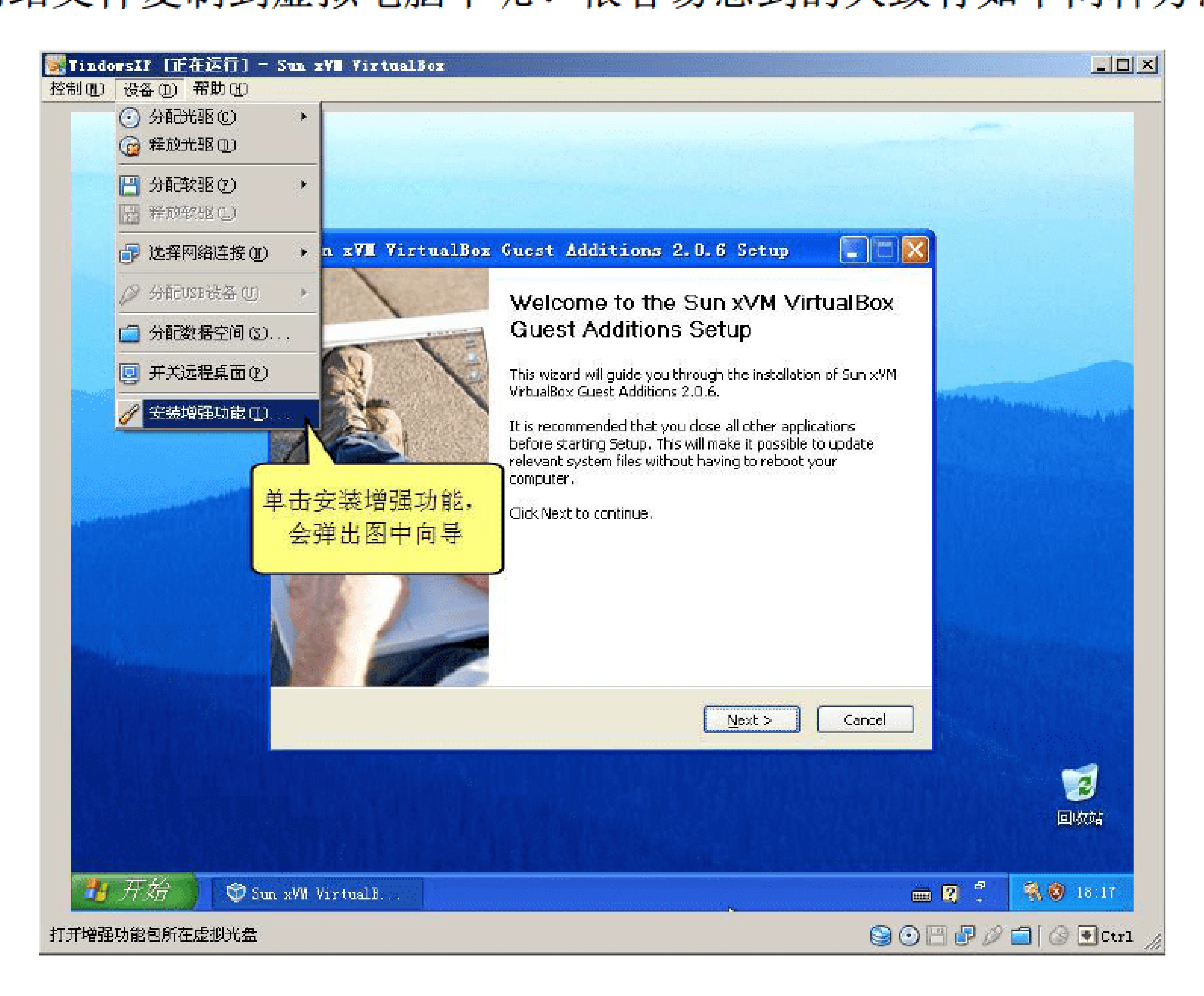


图 9-27　安装操作系统完毕后可安装增强功能

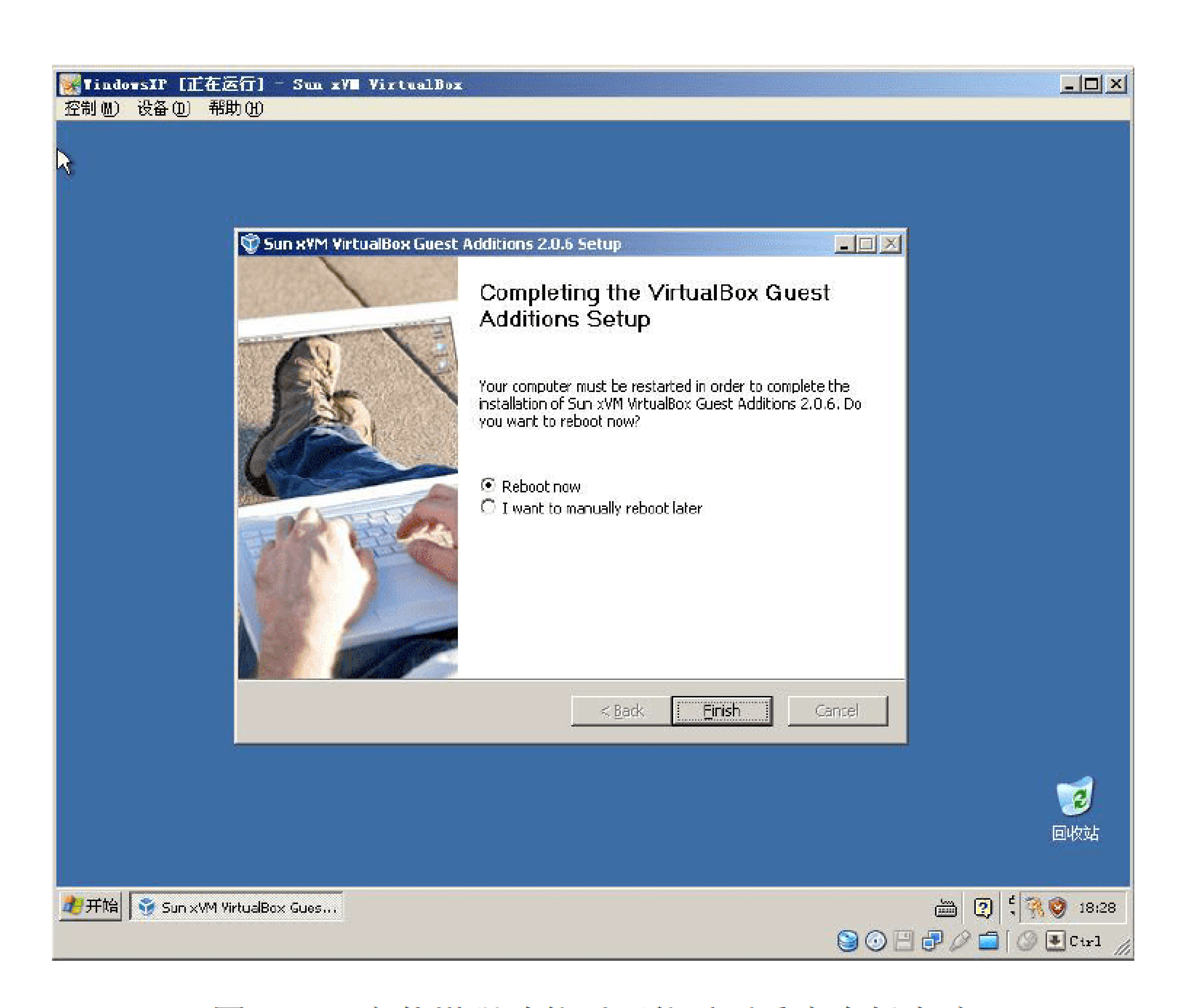


图 9-28　安装增强功能后可能需要重启虚拟电脑

- 利用网络：在其他电脑上直接访问该虚拟电脑的硬盘，将新代码复制到 Web 应用服务器管理的网站文件夹。
- 利用 VirtualBox 自身提供的数据空间功能实现网站文件的共享。本节将介绍这种方法。

在虚拟电脑的显示窗口右下方，我们可以注意到有一排很小的图标，这是当前虚拟电

脑的状态栏。单击其中的“文件夹”图标，会弹出数据空间窗口，就可以对虚拟电脑与宿主电脑之间的共享进行设置，如图 9-29 所示。

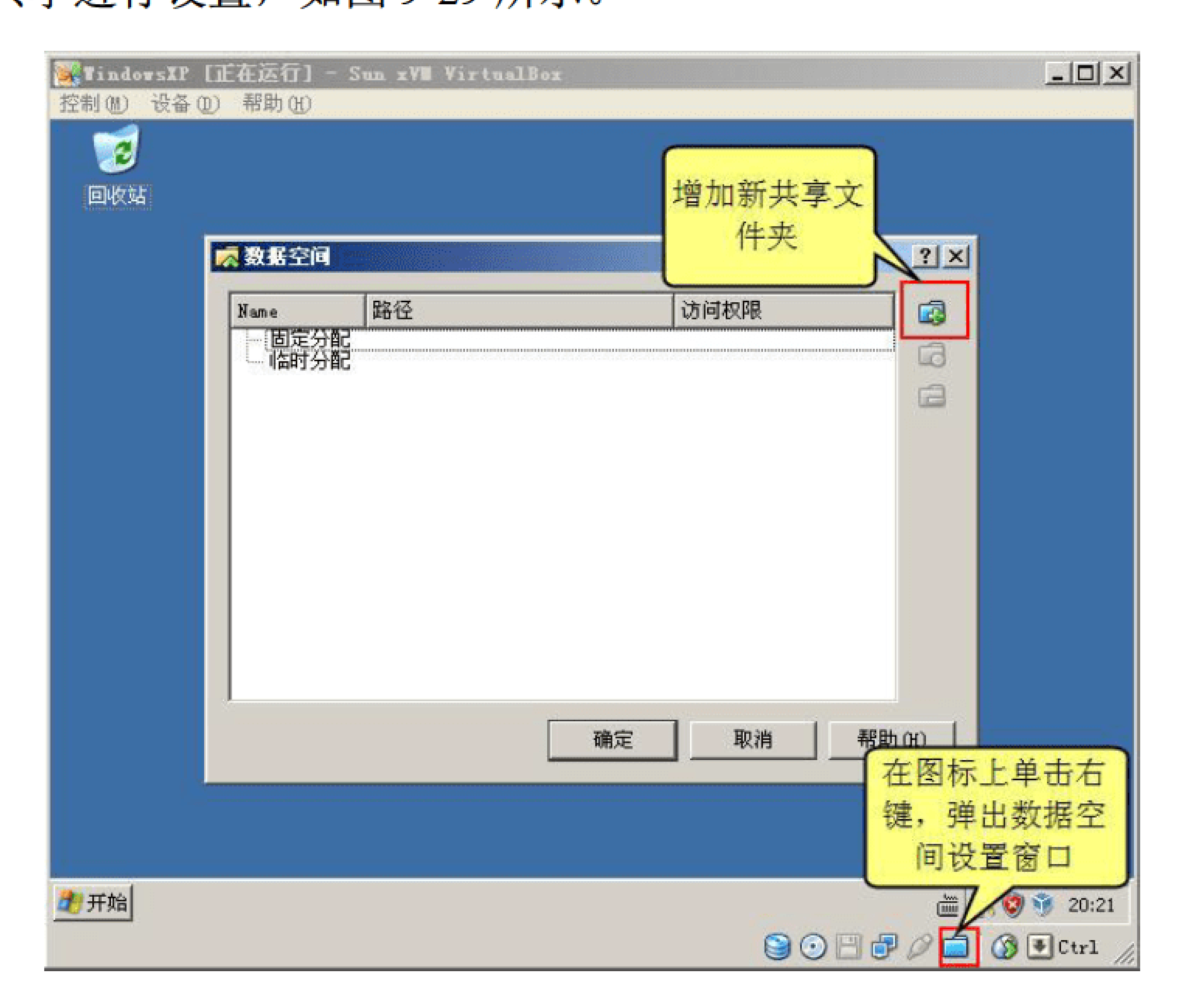


图 9-29　打开设置共享文件夹窗口

在图 9-29 数据空间窗体中单击右上方的新建文件夹图标，添加一个新的数据空间，如图 9-30 所示。首先需要找到在宿主电脑中存放待共享文件的位置，然后在 Folder Name 文本框中输入一些便于记忆的文字作为该数据空间的名称。

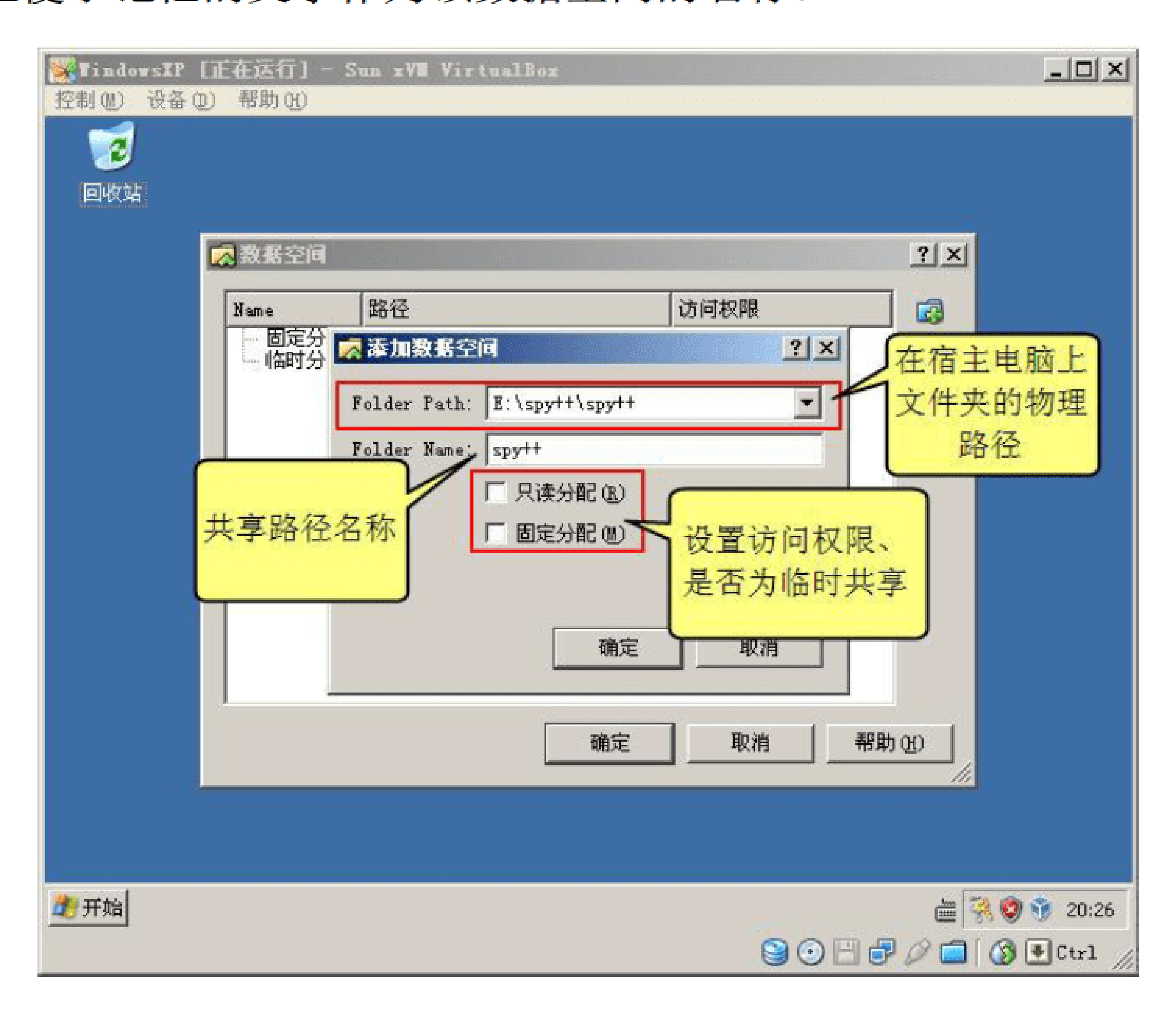


图 9-30　创建新的共享文件夹

【数据空间就是特殊的共享文件夹】

其实，VirtualBox 中的数据空间就是一种特殊的共享文件夹，只能用于宿主电脑与虚

拟电脑之间交换数据。在其他的电脑上，是看不到这些共享文件夹的。

数据空间增加完毕后，在虚拟电脑中如何打开呢？

方法很简单，使用虚拟电脑中的资源管理器，如图 9-31 所示，在“文件夹”区域中依次展开“网上邻居”|“整个网络”| VirtualBox Shared Folders 文件夹，就可以找到设置成功的数据空间列表。单击其中任意一个共享文件夹，就可以看到宿主电脑上存放的所需文件了。

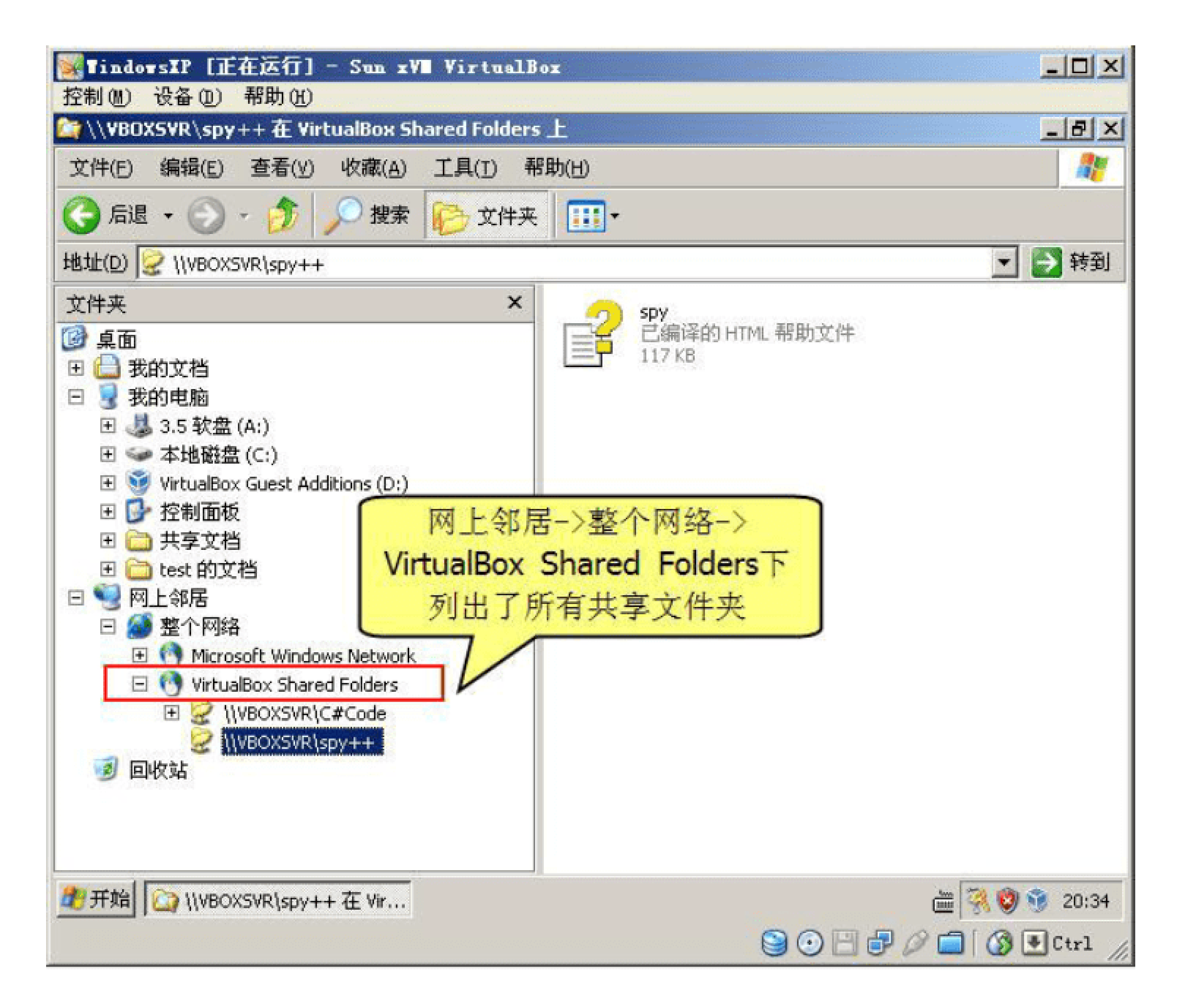


图 9-31　通过资源管理器找到共享的文件夹位置

9.3.6　利用 VirtualBox 组建网络

在实际的 Web 性能测试工作中，我们或者需要从不同操作系统不同配置的虚拟电脑上访问网站以测试速度，或者要模拟网站的生产环境，因此，网络对于虚拟电脑是必不可少的，本节中将对组建网络做简单的介绍。

VirtualBox 提供了 3 种在虚拟电脑中构建网络的方式。

- 默认的 NAT 方式：这种方式很简单，安装后不必进行很多的设置就可以访问互联网络，只要宿主电脑可以访问互联网络即可。这项功能对于性能测试中模拟网站访问的客户端是很有益处的。
- 主机界面方式（Host Interface）：这种方式能够构建出比较复杂网站生产环境中的网络。在此方式下，宿主电脑、其他物理电脑、虚拟电脑之间可以构建出互相连接的网络。
- 内部网络方式（Internal Network）：这种方式可以在多台虚拟电脑之间组建网络，但是，虚拟电脑与宿主电脑或者其他物理电脑之间是不相连通的。

设置虚拟电脑的网络可以通过如下方法来进行。

（1）在 VirtualBox 主界面中选择待设置的虚拟电脑。

（2）选择“管理”|“设置”命令，弹出当前虚拟电脑的设置窗体，如图 9-32 所示。选择其中的“网络”选项页。

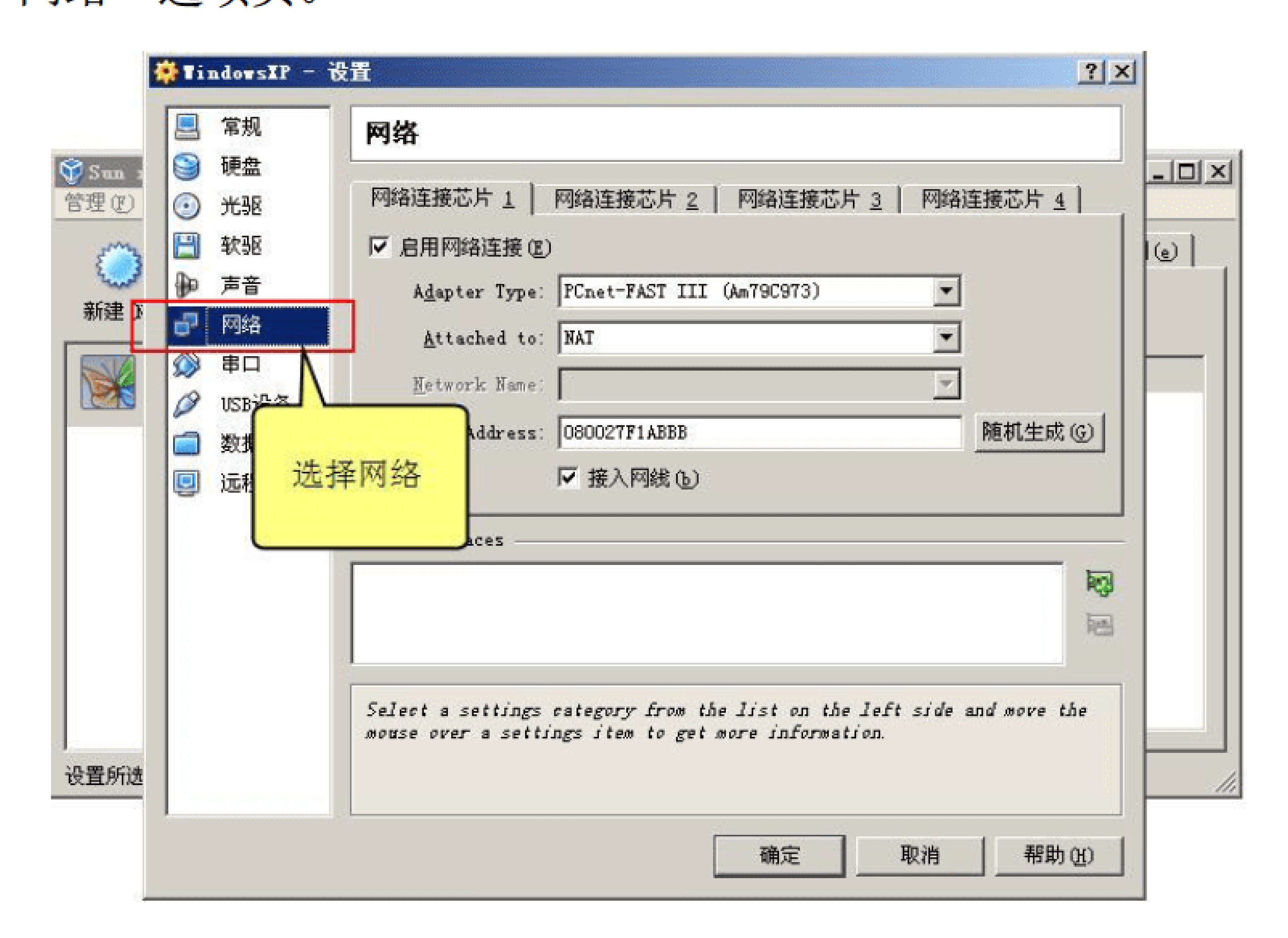


图 9-32　对虚拟电脑进行网络设置

（3）如图 9-33 所示，在 Attach to（连接到）下拉列表框中，依据需要选择 NAT、Host Interface 和 Internal Network 等 3 种方式，由于默认的 NAT 即可满足要求，所以一般情况下并不需要特别指定。



图 9-33　虚拟电脑中可以应用的 4 种网络类型

（4）按照图 9-34 的步骤增加一块网卡。单击“确定”按钮，重启虚拟电脑使设置生效。

（5）进入系统后，可以发现虚拟电脑已经可以连接到互联网络了，如图 9-35 所示。当然，这个前提是宿主电脑本身可以接入互联网络。

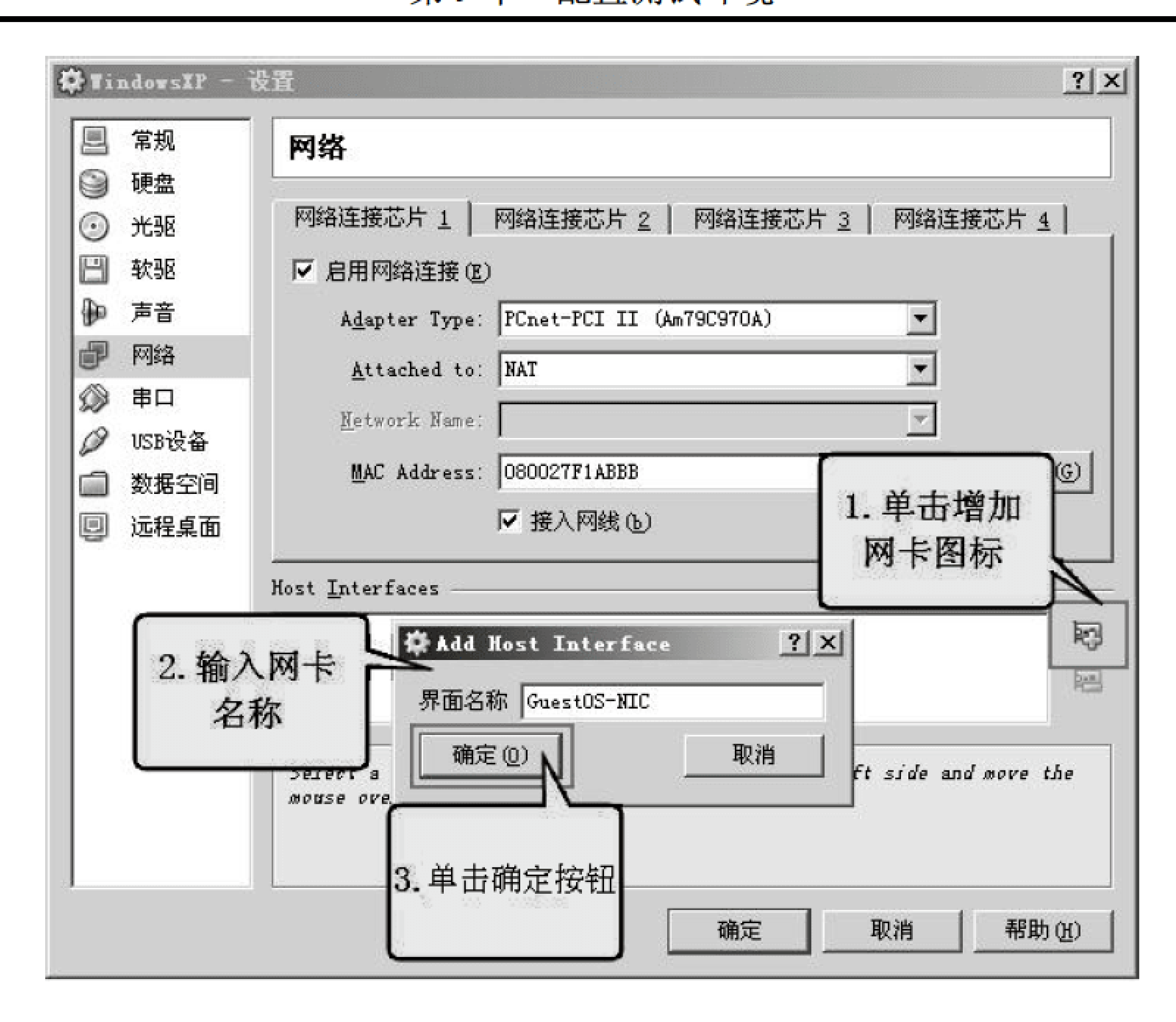


图 9-34　为虚拟电脑增加一个网卡界面



图 9-35　在虚拟电脑中浏览网页

对于 Host Interface 方式与 Internal Network 方式设置网络，由于与实际情况联系紧密，在本书中就不介绍了，感兴趣的读者可以参阅相关帮助文档进行设置。

9.3.7　VirtualBox 中的状态备份

VirtualBox 中的状态备份是很有用的一个功能，它就好比文本编辑器中的撤销操作，使得我们能够返回到虚拟电脑不同时间不同配置下的状态。设想这样一个应用场景：每天

测试工程师都需要用当天新生成出来的版本更新虚拟电脑中的旧版本产品。如果在这样的情况下，虚拟电脑首先回复到最初未安装产品、“干净”的状态，再进行新版本产品的安装，则可以省去卸载旧版本的时间和可能存在的一些问题，提高工作效率。

如图 9-36 所示为选择的虚拟电脑备份当前状态的界面。可以看到，在 VirtualBox 的主界面，选择处于窗体右半部分的“快速修复”选项卡，单击其中的“生成备份”按钮，在弹出的对话框各文本框中分别输入备份名称和描述，即可生成包含当前状态的备份文件。

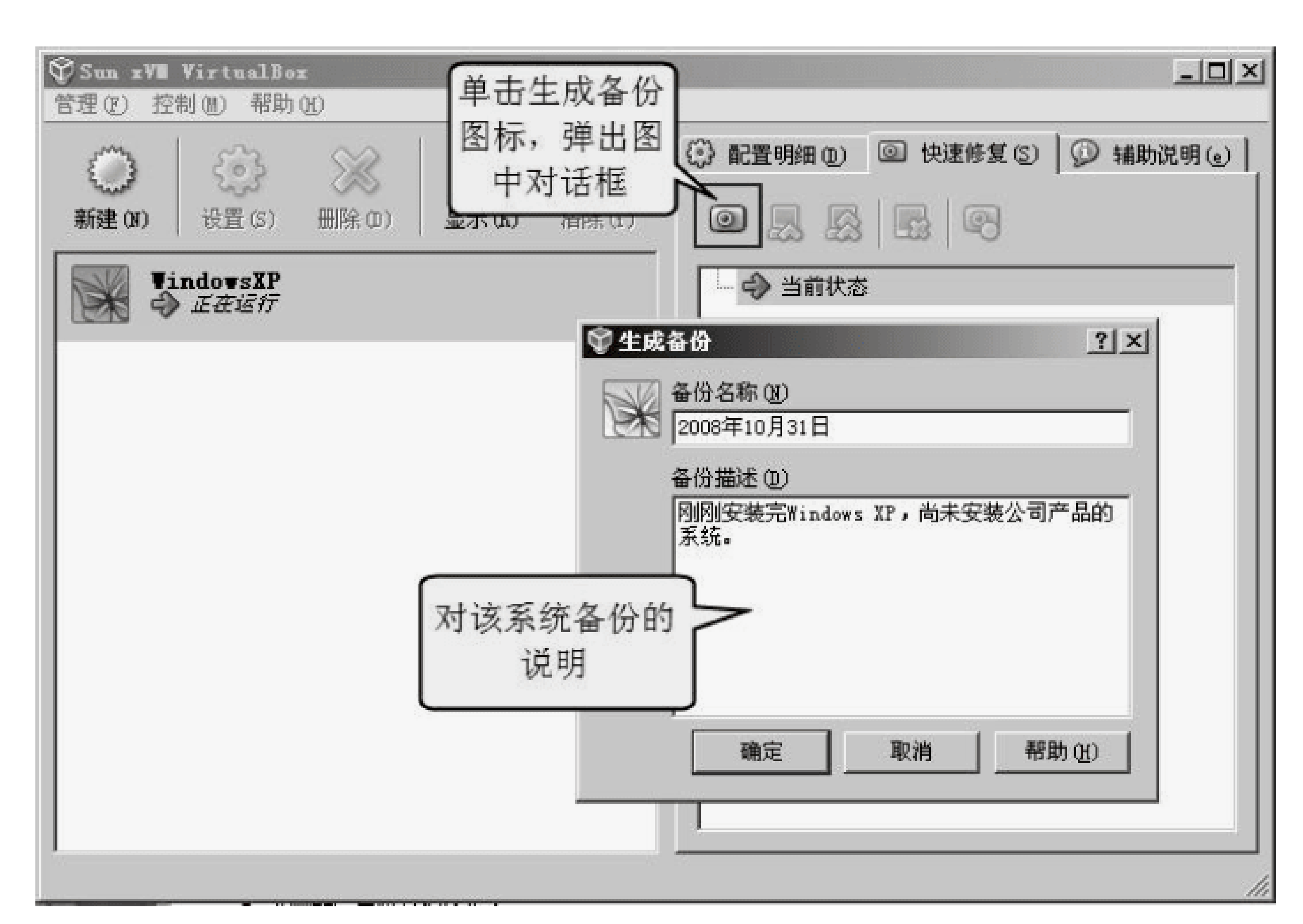


图 9-36　为虚拟电脑生成新的备份

在生成了多次备份之后，回复到其中某一个备份也是很容易的，只需要参照图 9-37 中的各个工具栏图标进行操作即可。

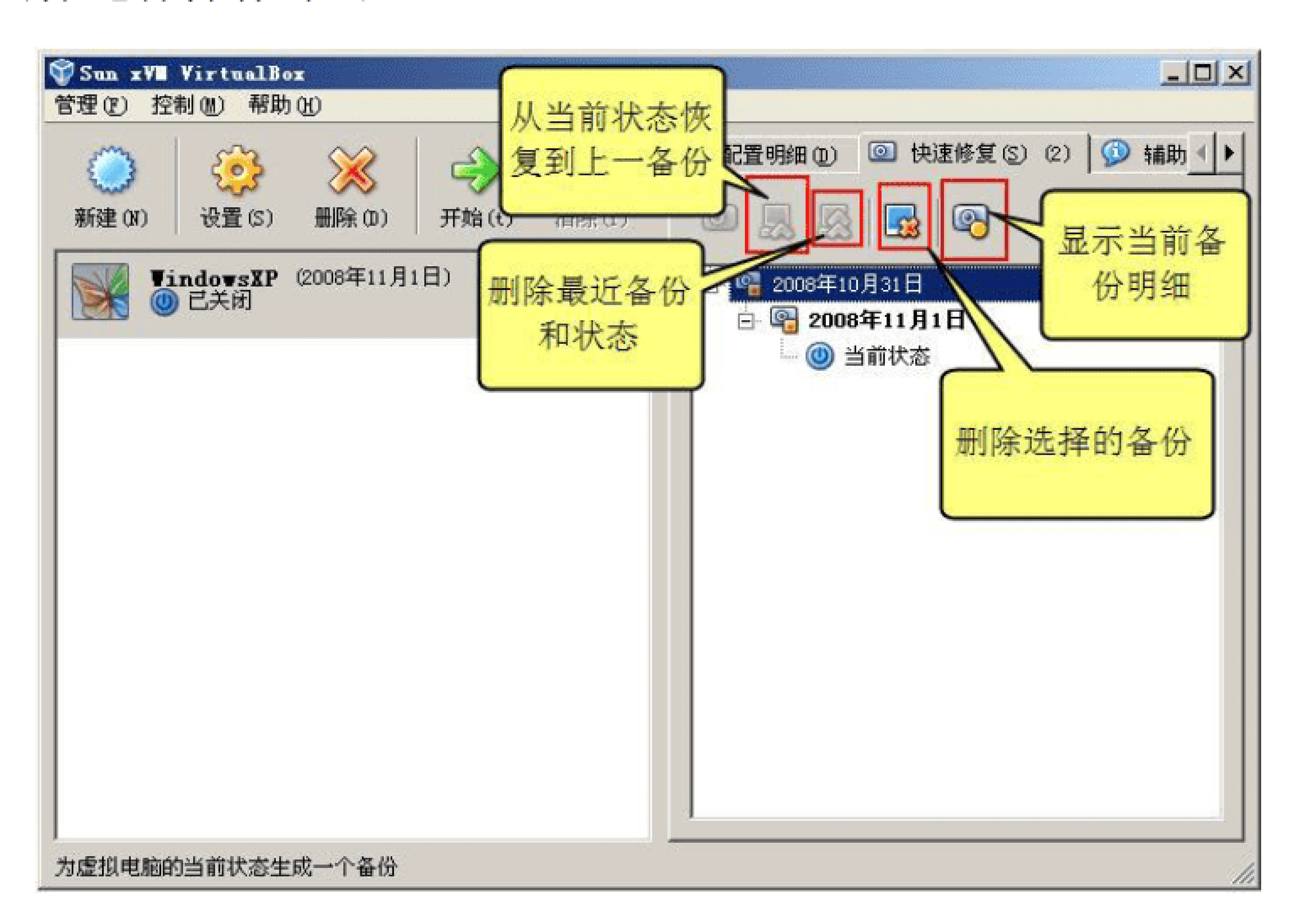


图 9-37　对于备份的操作

【VirtualBox 的不足】

不过需要注意的是，VirtualBox 在备份状态方面还存在一些不完美的地方：它只能按照“原路返回”的方式，而无法恢复其他备份分支的状态，这种情况如图 9-37 所示。对于

其他的虚拟机软件，则有所不同，比如微软公司出品的 Hyper-V，可以随意地恢复到不同分支下的状态，只不过在 Hyper-V 中，备份被称为“快照”。读者在选择虚拟机软件的时候，要根据自己的需要来进行。

对于其他的虚拟机软件，功能与 VirtualBox 大致相同：都包含功能增强插件；都支持网络的配置；都支持程度不同的状态备份等。本书对于 VirtualBox 的介绍相对简单和笼统，读者还需要在实际工作中不断地积累经验，让虚拟化技术更好地为测试，特别是性能测试工作服务。

9.3.8　使用 VirtualBox 搭建测试环境

在前面的章节中，我们利用 VirtualBox 成功地创建了一台 WindowsXP 系统的虚拟电脑。在 Web 应用的性能测试方面，WindowsXP 一般都是用作客户端，用于验证相应时间，浏览器兼容性等方面。实际上，虚拟化软件完全可以虚拟提供 Web 应用的服务器，所不同的是需要在虚拟电脑上安装服务器版操作系统、安装应用服务器、数据库服务器等，还要安装 Web 应用的代码以供调试。

使用虚拟化软件的备份或者快照功能可以很快恢复机器的不同状态。在测试过程中，这项功能对于需要反复调试程序代码或者更改相关服务器的设置非常方便。我们可以在虚拟电脑应用某种配置组合之前和之后都保存一个状态，当测试完毕、需要进行下一种配置组合的测试时，只简单地恢复到最初状态再进行修改即可。

业内比较普遍的经验是，测试部门拥有一个专门的文件服务器用来储存装好各种操作系统的虚拟电脑，一旦测试工作需要，可以直接将它们复制出来，通过虚拟化软件进行管理，从而快速搭建出一个合格的测试环境。

9.4　本 章 小 结

本章首先介绍了测试环境以及准备测试环境的重要性。在整个过程中，要注意如下原则：

- 尽可能地模拟真实生产环境。
- 在满足第一条的同时，利用一些技术比如虚拟化减少环境配置时间，提高真正用于测试的时间比例。

在实际工作中，可以通过虚拟化技术快速地搭建测试环境，虚拟化技术的优点主要有：

- 快速完成不同配置系统的切换，从而节省部署操作系统、配置生产环境的时间，增加真正用于测试产品的时间，提高工作效率。
- 在不清楚产品对于某个平台的支持情况时，可以先利用虚拟化软件进行模拟，这样可以预先发现和了解在该平台下真正执行测试过程中可能遇到的困难和解决办法。

在本章的最后部分，介绍了 Sun 公司的 VirtualBox 虚拟化软件的主要功能。市场上的虚拟化软件产品很多，而且大多数都是免费使用的，读者可以根据实际需要选择不同的虚拟化软件，以实现前述提高工作效率的目的。

第 10 章　LoadRunner 中的场景

在前面的章节里，小白已经熟悉了 LoadRunner 的脚本如何录制、修改和回放。实际上，脚本描述的是一个虚拟用户的行为。仅仅模拟一个虚拟用户的访问，对于测试 Web 应用的性能是基本没有意义的，真正有价值的数据需要通过大量用户的模拟才能获得，而这一步骤则需要本章介绍的知识：场景。

场景是对大量真实用户操作规律的模拟，它包含了执行规则与用户数量的定义。LoadRunner 的场景是在控制器程序（Controller）中定义的。控制器可以通过以下途径打开：单击“开始”|“程序”| LoadRunner | LoadRunner 命令，弹出程序启动界面，如图 10-1 所示。需要运行控制器的时候，在图中单击框住的链接 Run Load Tests（运行性能测试）即可。

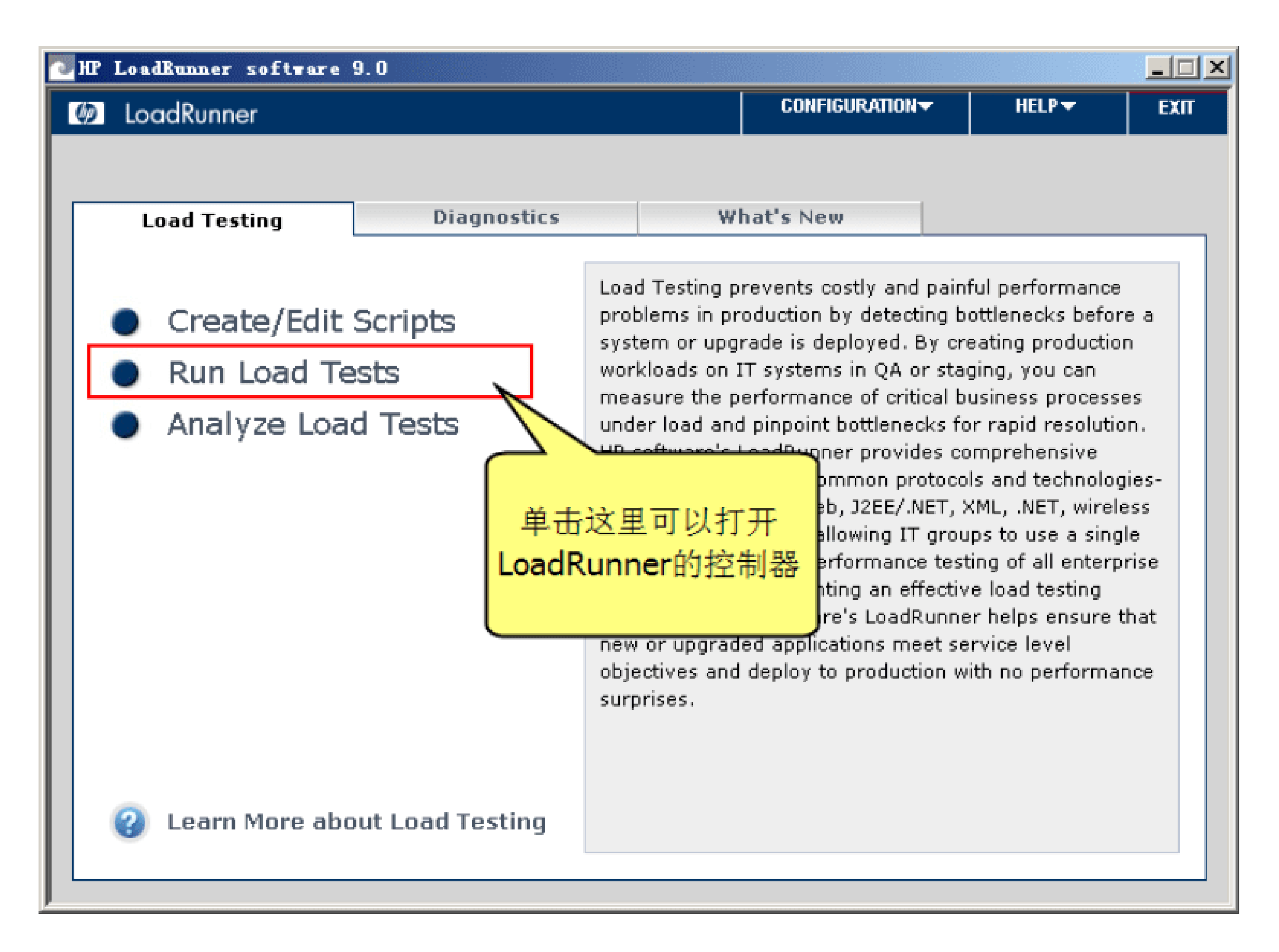


图 10-1　运行 LoadRunner 的控制器程序

本章将对场景的诸多设置进行详细讲解。

10.1　场景的创建

打开场景创建对话框有两种途径，分别对应两种不同的情况：

- 单击图 10-1 方框中链接，打开控制器程序，可以为新建的场景选择多个执行脚本。
- 在图 10-2 中，选择虚拟用户（VuGen）程序中的 Tools|Create Controller Scenario

（工具 | 创建控制器场景）命令为当前的脚本设置场景。

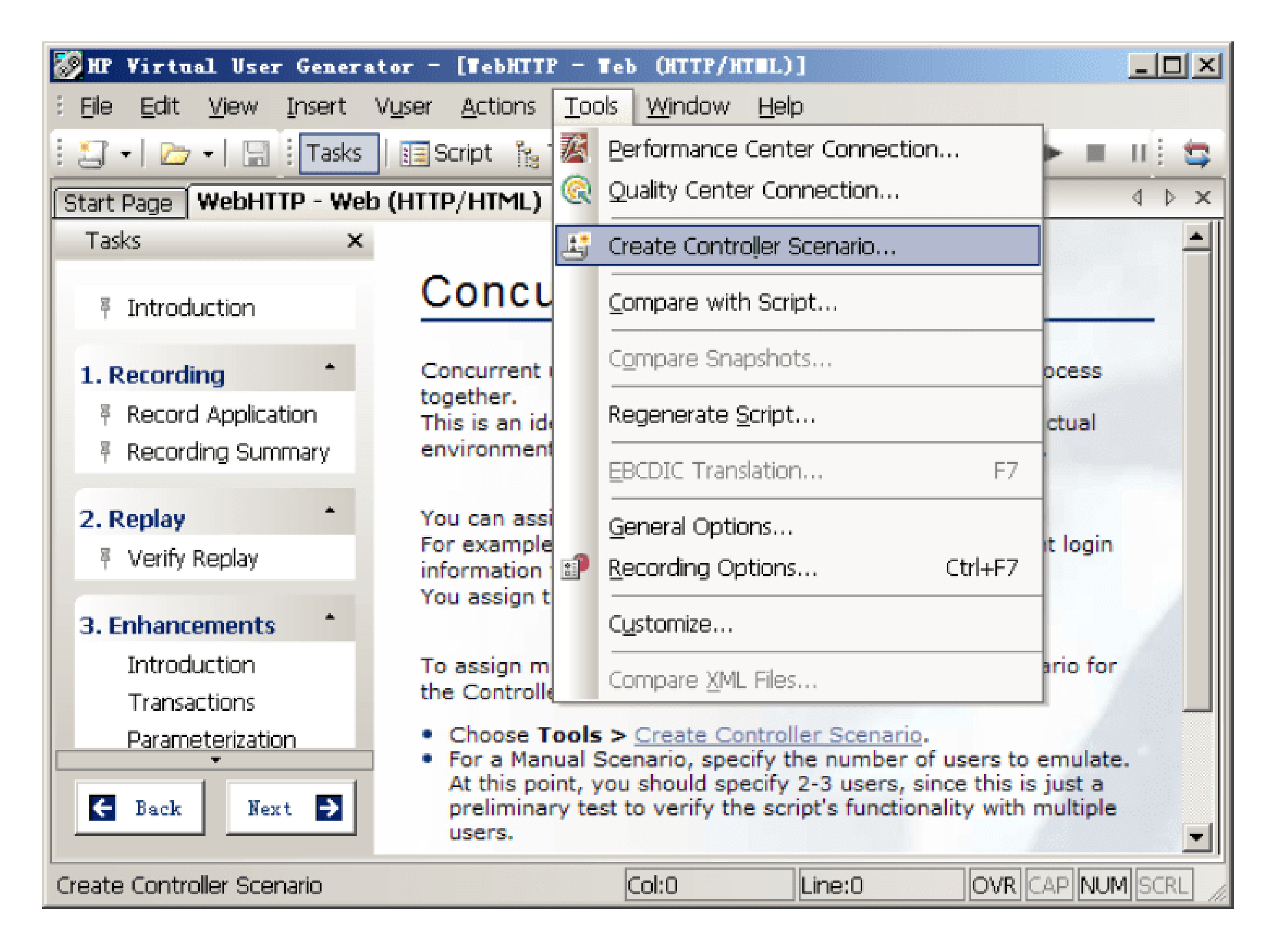


图 10-2　在录制脚本界面可以为当前的脚本创建场景

由于场景是大量虚拟用户访问 Web 应用的规则，每个虚拟用户执行的脚本一般不限于一个，因此，在实际工作中，使用第一种方法是更为普遍的。

10.1.1　场景创建设置对话框

前文提到，有两种打开场景设置对话框的方法，因此，这两个方法打开的设置对话框界面也有所不同，如图 10-3 和图 10-4 所示，读者可以发现它们的区别。

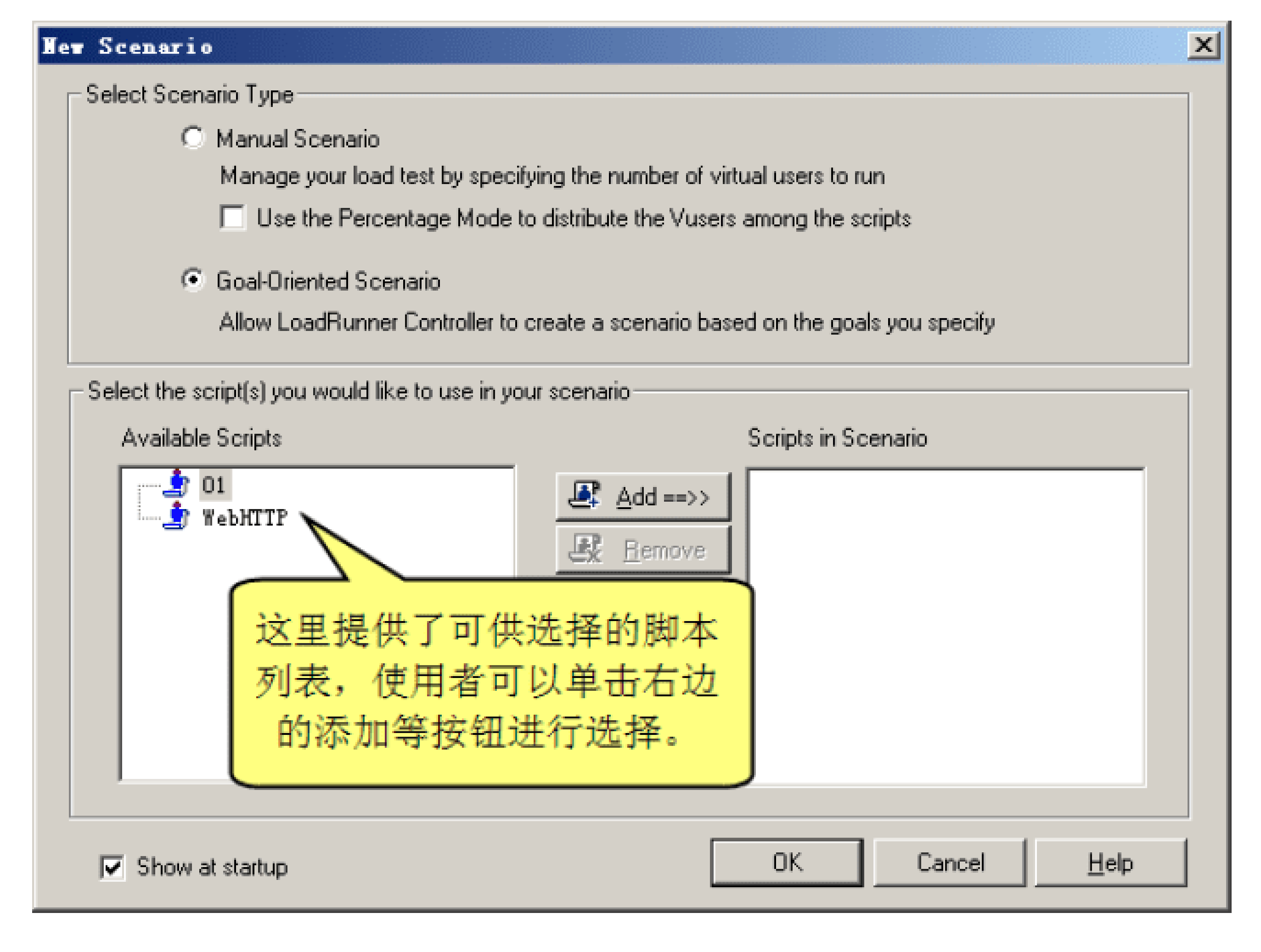


图 10-3　通过运行控制器打开



图 10-4　从虚拟用户生成器（VuGen）中打开

在图 10-3 中，设置对话框上半部分 Select Scenario Type（选择场景类型）选项区域列出了场景的两种类型：Manual Scenario（人工场景）与 Goal Oriented Scenario（面向目标场景）。而窗体的下半部分则是 Available Scripts（可用脚本）列表框：由于 Web 应用比

较复杂，在实际工作中需要创建一系列的脚本，比如登录脚本、订票脚本、回复帖子脚本等。因此，可以通过选择不同的脚本组合来模拟不同虚拟用户的不同操作。

从虚拟用户生成器（VuGen）中打开的创建场景对话框则相对简单，这是因为之前完成的脚本已经默认增加到待创建的场景当中了，因此不必提供脚本选择界面。这种方法并不需要打开控制器程序。

具体使用哪一种方法打开创建场景对话框，完全取决于工作中的实际情况与方便与否。

10.1.2　场景的分类

我们首先解释一下场景创建对话框的具体内容，为了讲述方便，以图 10-3 为例。前文提到图 10-3 窗体上半部分是设置场景分类的，那么，什么是人工场景、什么又是面向目标场景呢？

【人工场景】

所谓人工场景，实际就是自定义模式，各因素完全由我们来设置的创建场景方法。相比面向目标场景，人工场景在实际工作中应用的更为广泛。用赛车游戏来比喻，这种方法类似常规比赛，不同的汽车从同一起点出发，到同一终点结束，最终按照时间排出名次。

【面向目标场景】

面向目标场景则与人工场景有所不同，它预先定义了一个测试目标，LoadRunner 将根据这个目标自动构建场景，有点类似向导模式。这种方法对于验证在项目性能说明书中列出、需要达到的性能目标很方便。还是用赛车游戏来比喻，面向目标场景有点类似计时赛或者追逐赛，不同的汽车从同一起点出发，在规定的时间内，走的最远者获胜。

10.1.3　面向目标场景的创建

前文提到面向目标场景类似常用软件中的向导模式，它设定了一个或者多个测试目标，比如要求系统达到每秒处理 5 个事务，LoadRunner 再根据这些目标自动创建场景。目前，LoadRunner 支持的测试目标有如下几种：

- 虚拟用户数量。
- 每秒点击次数（只对 Web VUser 有效）。
- 每秒事务数量。
- 每分钟访问页面数量（也仅对 Web VUser 有效）。
- 事务响应时间。

在图 10-3 或者图 10-4 中分别选择 Goal Oriented Scenario（面向目标场景）单选按钮，添加任意脚本，都可打开 LoadRunner 的控制器（Controller），如图 10-5 所示。本书对于场景的操作都是在控制器中进行的，它与虚拟用户生成器（VuGen）一样，可以单独从程序组中启动。

在图 10-5 的左下方，系统默认选择了场景目标为：每秒点击次数 100。这样，虽然我们还没有进行自定义的设置，但也可以明白运行当前场景，能观察到当系统每秒点击次数达到 100 之前整个过程的性能情况。

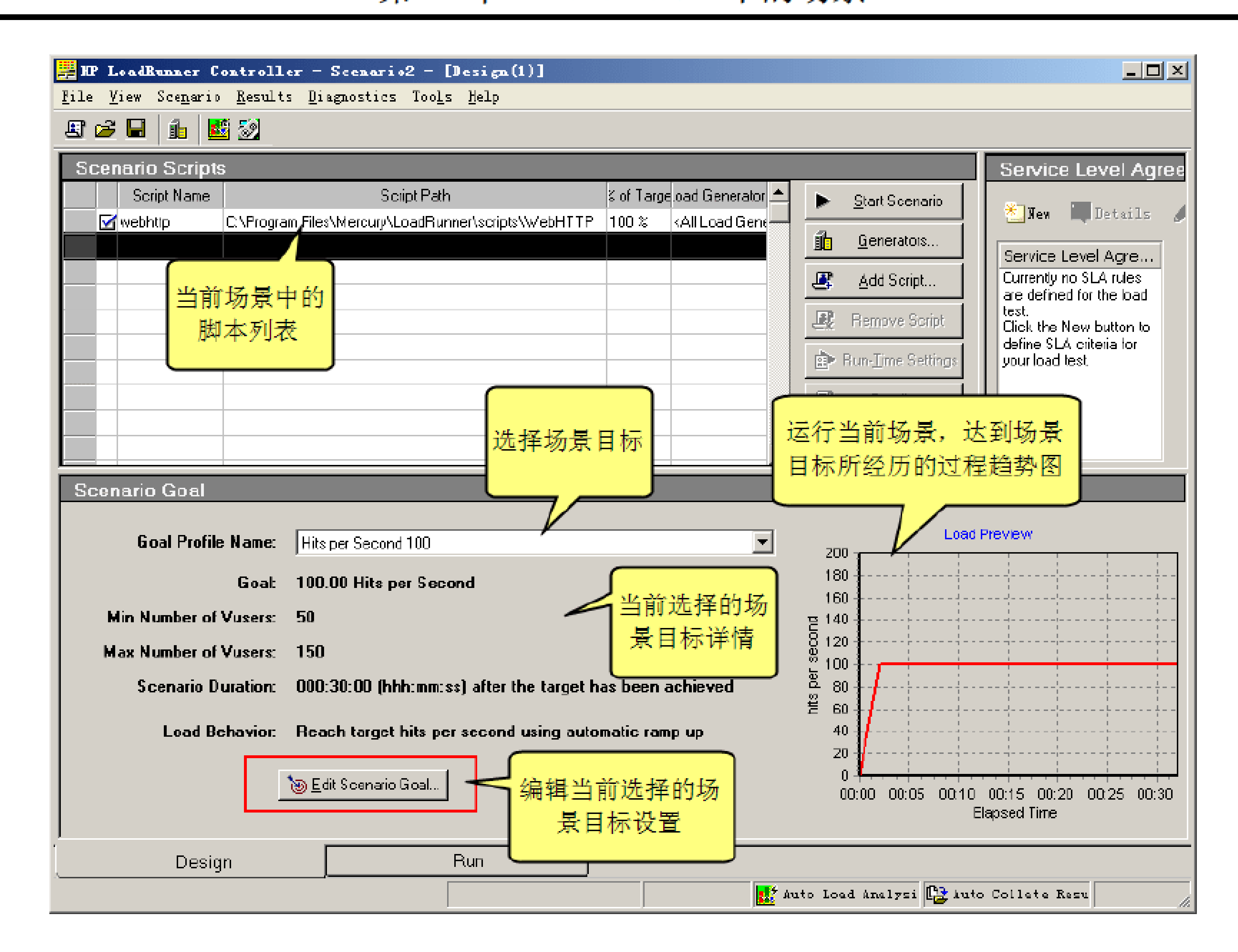


图 10-5　LoadRunner 控制器的界面

如果单击导航条或者菜单中的保存按钮，当前场景就创建完成了，系统会保存场景到一个后缀名为.lrs 的文件，默认在安装目录的 Scenarios 子文件夹下。

10.1.4　场景目标的编辑

在实际工作中，不可能只利用默认的场景设置，因此，有必要针对各个目标都进行一番编辑。对于后面要提到的手动场景，更要利用编辑场景的功能进行设置。

1．编辑场景目标

在图 10-5 中单击 Edit Scenario Goal（编辑场景目标）按钮，弹出对话框如图 10-6 所示。

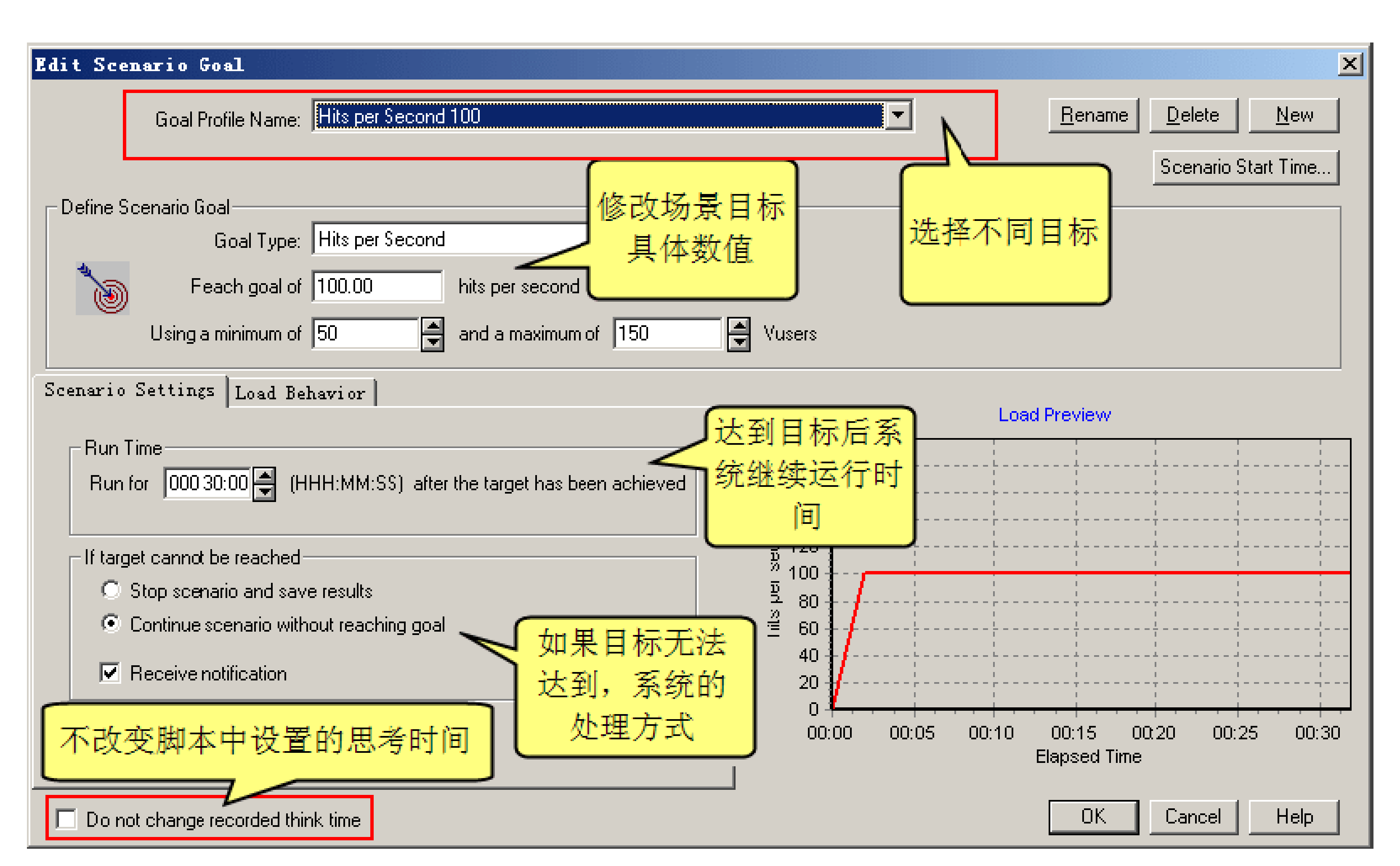


图 10-6　编辑场景目标

除了第一个目标：Virtual Users（虚拟用户数量）之外，其他目标都需要设置最大虚拟用户数和最小虚拟用户数量。这样一个给定的用户数量范围是为了更真实地模拟实际情况。真实情况下，每个用户的操作都是不同的，有的操作可能是多个用户同时点击（比如登录系统），有的操作则可能是少量用户连续快速多次点击（比如游戏中的单击操作），因此有必要根据实际情况进行调整，多次运行，从结果中发现可能的性能变差规律。

【达不到目标的处理方式】

肯定有读者会问，如果达不到设定的目标，LoadRunner 将如何处理呢？这种情况也是可能发生的，特别是在 Web 应用刚开始测试的阶段，我们对其性能表现并不熟悉，因此制定的测试目标，比如每秒点击数量数值很可能过高，无法达到。在这样的情况下，也可以通过图 10-6 左下部分进行设置：选择停止场景并保存结果或者继续运行场景、无须达到目标，另外，还可以选中接收通知使得测试人员了解测试目标无法达到这一情况。

在图 10-6 中，还可以通过选择 Load Behavior（负载行为）选项卡设置为达到当前目标而增加负载的方式，其界面如图 10-7 所示。

负载增加的行为方式有 3 种，分别如下：

- ❑ 自动（Automatic），这是默认方式，无须设置。
- ❑ 时间间隔（Reach target after），这种方式可以设置当前场景在达到目标之前需要运行多长时间，以小时：分钟：秒为单位。
- ❑ 渐进式（Step up by every），这种方式可以采取一种渐进增加的策略执行场景，比如图 10-7 中是每隔 2 分钟增加 20 个虚拟用户。其他的目标具体设置内容和数值有所不同。

2. 场景的执行方式

当有关场景目标数值、负载增加方式、达不成目标的后处理均编辑完毕之后，还可以设置场景的启动时间。方法是单击图 10-6 中右上方的 Scenario Start Time（场景开始时间）按钮，弹出对话框如图 10-8 所示。

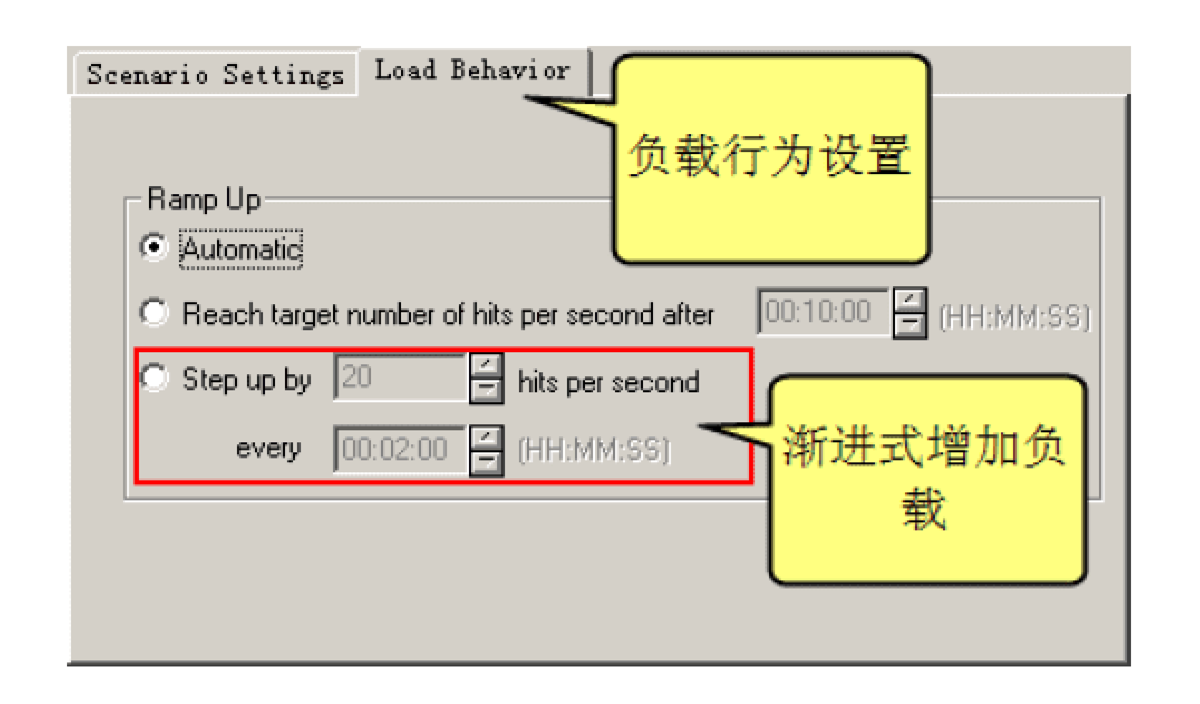



图 10-7　设置达到场景目标前的负载行为

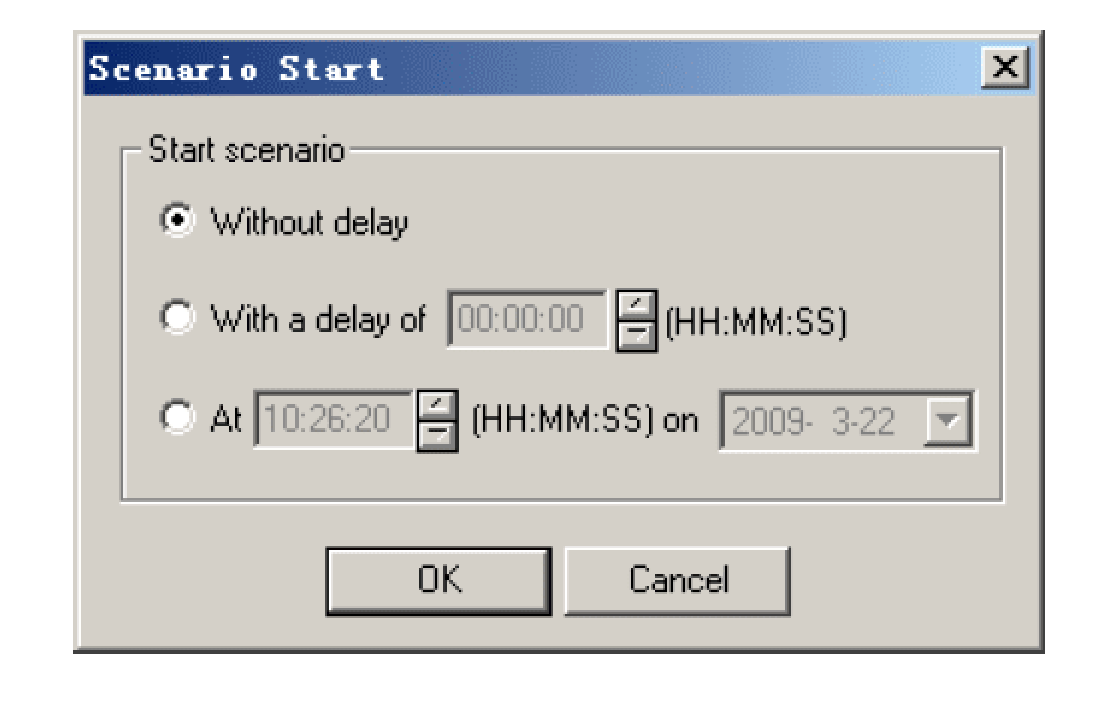


图 10-8　设置当前场景的开始时间

在图 10-8 中，可以设置如下 3 种方式执行当前场景：

- ❑ 立刻执行（Without delay）。
- ❑ 延时执行（With a delay of）可以设置具体时间之后再运行场景。
- ❑ 定时执行（At）可以设置在何时（具体日期、小时）运行场景。

【思考时间】

在图 10-6 中，我们还可以发现 Don’t change recorded think time（不修改录制的思考时间）选择框。前文提过，思考时间是用户在 Web 应用各操作之间的时间。因此，在与事务相关的场景目标设置中，若维持每秒事务数量不变，如果选中了此项，则虚拟用户数量要相应增加。

10.1.5　手动场景的设置

前文我们已经了解到场景与脚本相配合，能够代表 Web 应用运行时的真实状况。场景模拟用户访问的分布规律，脚本模拟用户的实际操作。多数情况下，在实际工作中更为有用的是执行手动场景的测试。手动场景也可以称为自定义场景。

手动场景可以选择两种方式来设置，分别是用户组方式和分布百分比方式。用户组方式将虚拟用户分组，测试工程师可以自由地分配各组用户数量；分布百分比方式则需要测试工程师指定某些用户所占的百分比和用户总数，系统再根据这些数值计算产生具体某类用户的数量。从这两类方式可见，它们都是对用户进行分配的，只不过出发点不同。LoadRunner 默认是用户组方式。

在图 10-3 或者图 10-4 中选择 Manual Scenario（手动场景）单选按钮，即可弹出 LoadRunner 控制器运行场景的界面，如图 10-9 所示。在图的最下方，有两个选项卡，分别是 Design（设计）和 Run（运行）。它们清楚地描述了手动场景的设置步骤就是：先设计，再执行。而设计手动场景的第一步，就是增加上面所述的用户组。

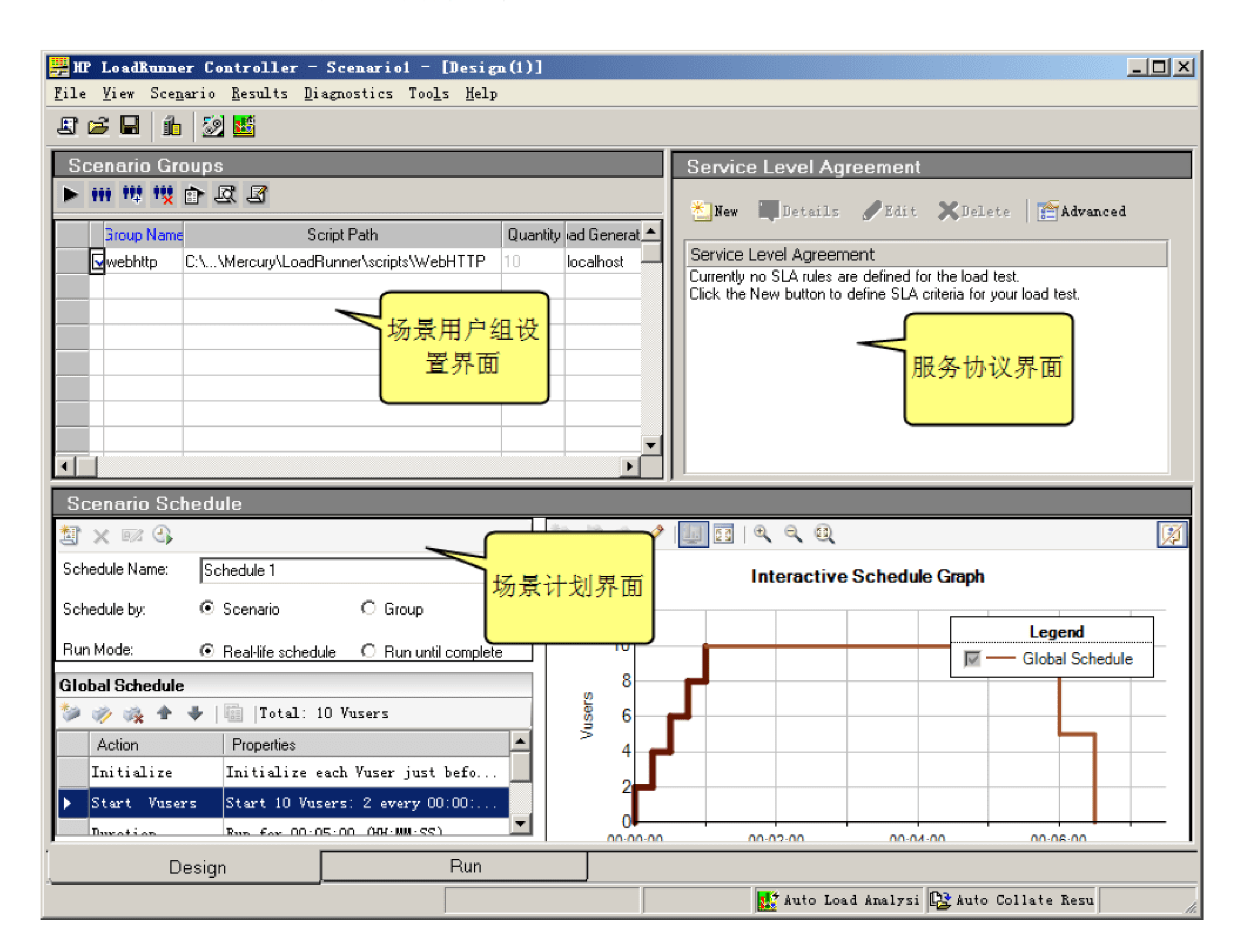


图 10-9　控制器中手动场景设置界面

【用户组】

所谓用户组，实际就是一群虚拟用户，它们要对被测试软件的操作是一致的。LoadRunner 通过一个或者多个用户组对被测软件的访问来考察性能。

测试工程师可以通过单击 Scenario Groups（场景用户组）视图上方的 Add User Group（增加用户组）等图标按钮进行用户组管理，如图 10-10 所示。

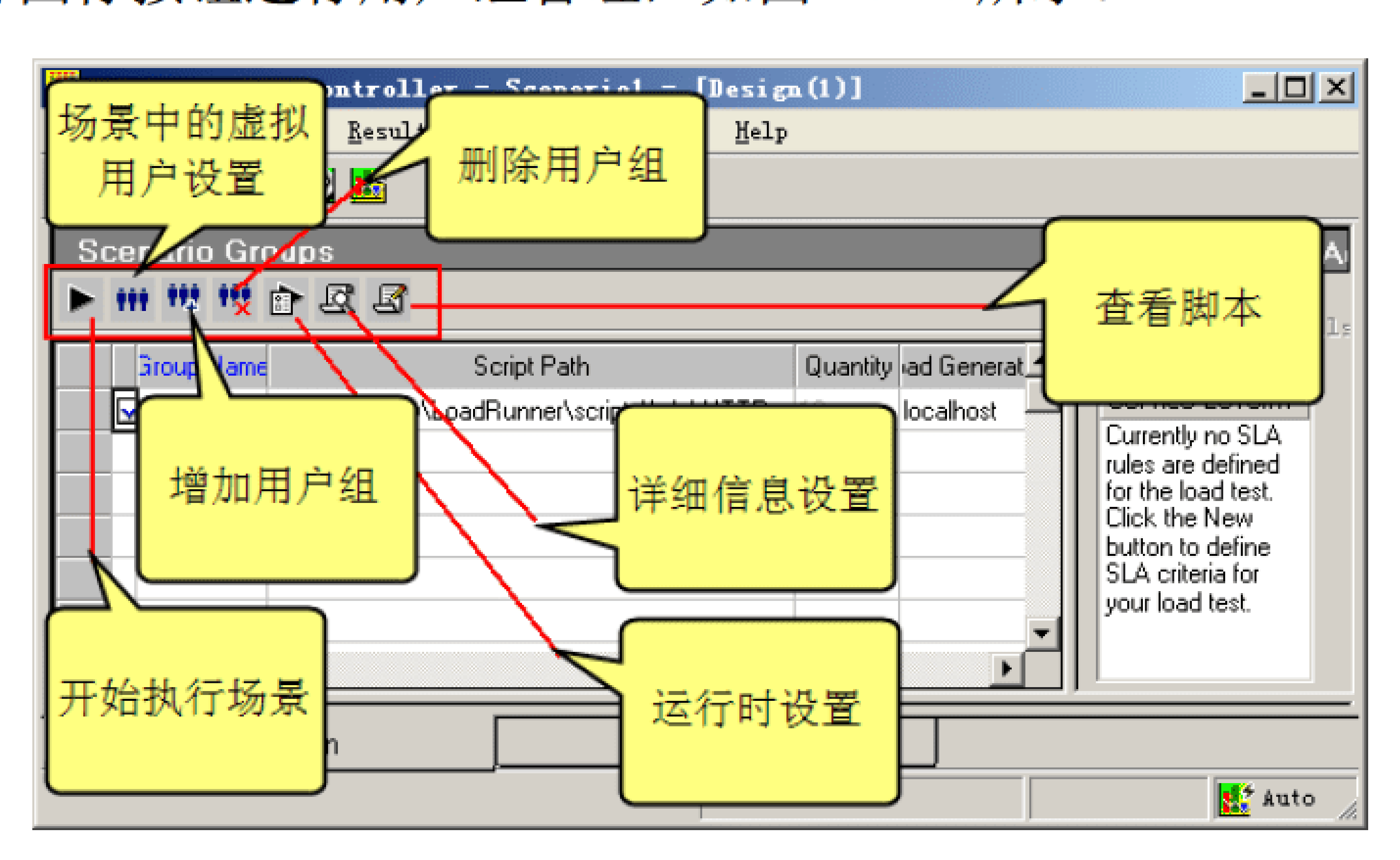


图 10-10　对场景用户组进行管理的工具条

本节将重点讲述用户组方式。实际上，分布百分比方式与用户组是类似的，都是通过不同的途径来描述场景中虚拟用户的分布规律，设置它们在执行场景时的行为方式。用户组与分布百分比两种方式之间还可以相互转换，如图 10-11 所示。

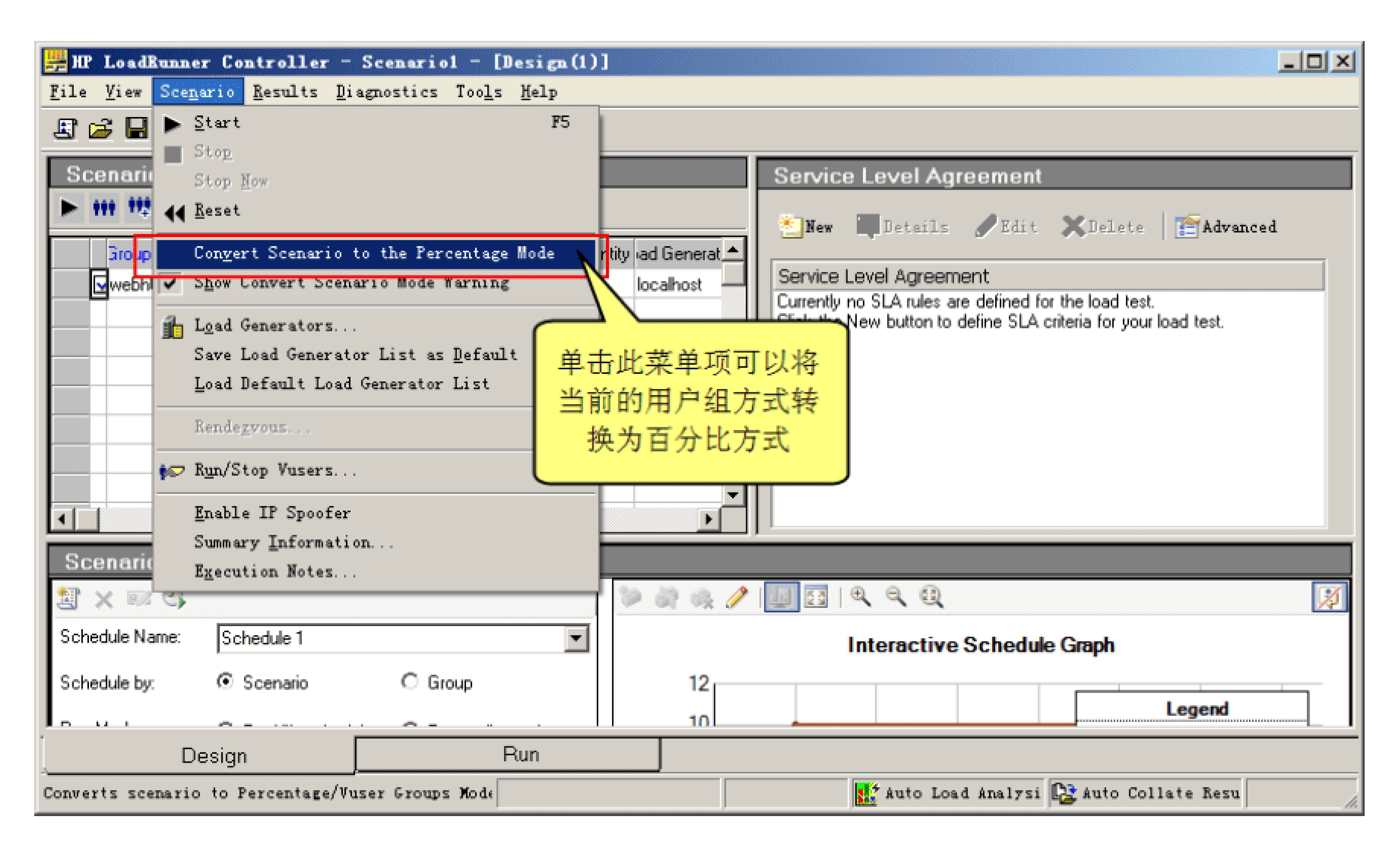


图 10-11　用户组方式与百分比方式的转换

另外，如果在打开控制器时选择 Use the Percentage Mode to distribute the Vusers among the scripts（在脚本中对虚拟用户分布使用百分比方式）选择框，也可以直接采用这种方式，如图 10-12 所示。

在 10-9 中，还可以为不同的用户组指派运行场景时执行不同的脚本，这样做是很有必要的：在真实的环境中，不可能所有的浏览者都在一个页面或者进行相同的操作，他们可能有的在注册新用户，有的在浏览商品信息，有的在回复留言，有的在上传图片等，因此

将进行不同操作的用户分组也是一个好办法。

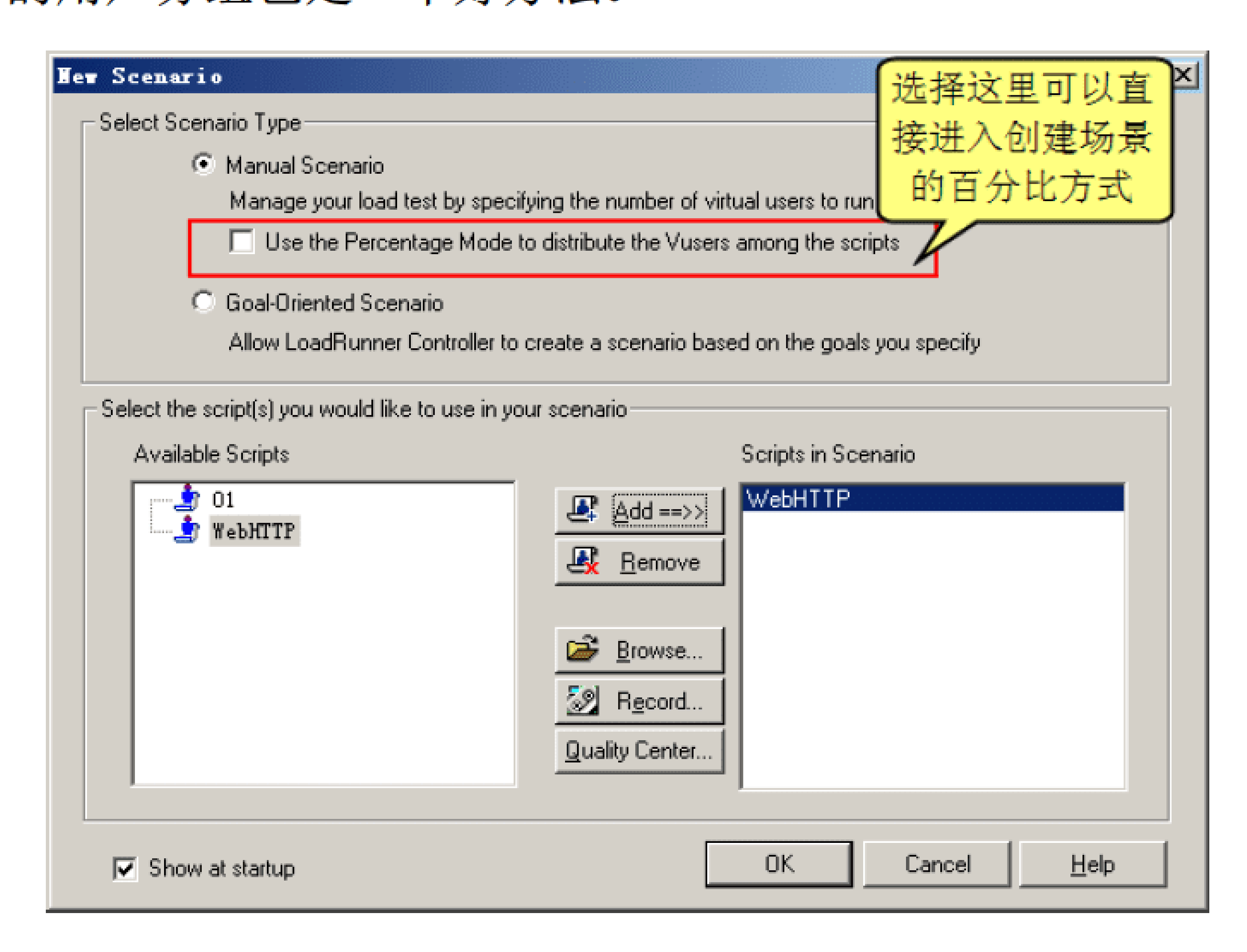


图 10-12　在创建场景时选择百分比方式

【多组同脚本和同组多脚本】

多组同脚本就是多个用户组使用同一个脚本执行场景。这对于测试 Web 应用的某一个特定功能是很有用处的，多见于 Web 应用的性能测试初期，由于被测试的功能范围较小，利于发现问题。同组多脚本则是同一个组的虚拟用户运行多个脚本，多见于 Web 应用的性能测试后期，以观察整个 Web 应用的性能情况，有点类似于集成测试，查看多个页面之间交替运行的性能情况。

10.1.6　压力产生器

在图 10-9 中，可以注意到在脚本列表后边都有压力产生器（Load Generators）这样的标识。这是 LoadRunner 为了更为真实地模拟实际情况而设置的：在 Web 应用上线运行之后，使用者会通过不同的客户端电脑进行访问。对这一特点的模拟就是通过 LoadRunner 控制器中的压力产生器来实现的。它可以控制多台计算机，通过其上虚拟用户的操作，对被测试的 Web 应用施加压力。

1. 压力产生器的添加

单击图 10-9 中 Scenario Groups（场景用户组）视图上方导航栏中的 Load Generators（压力产生器）图标按钮，即可打开压力产生器设置对话框，如图 10-13 所示。

在图 10-13 中，可以通过界面右边各种按钮对压力产生器进行管理。首先，读者需要通过单击 Add（增加）按钮创建一个压力产生器，如图 10-14 所示。

在 Name（名称）文本框中填入机器名称或者 IP 地址，在 Platform（操作系统）下拉列表框中选择那台计算机的系统类型（目前，LoadRunner 支持 Windows、Unix 两种）。对于 Temporary directory（临时目录）文本框，可以保持默认或者输入一个自定义的值。LoadRunner 的控制器将在临时目录中生成一些文件，用于储存场景运行时的数据。最后，

选中 Enable load generator to take part in the scenario（令压力生成器在场景中生效）复选框。

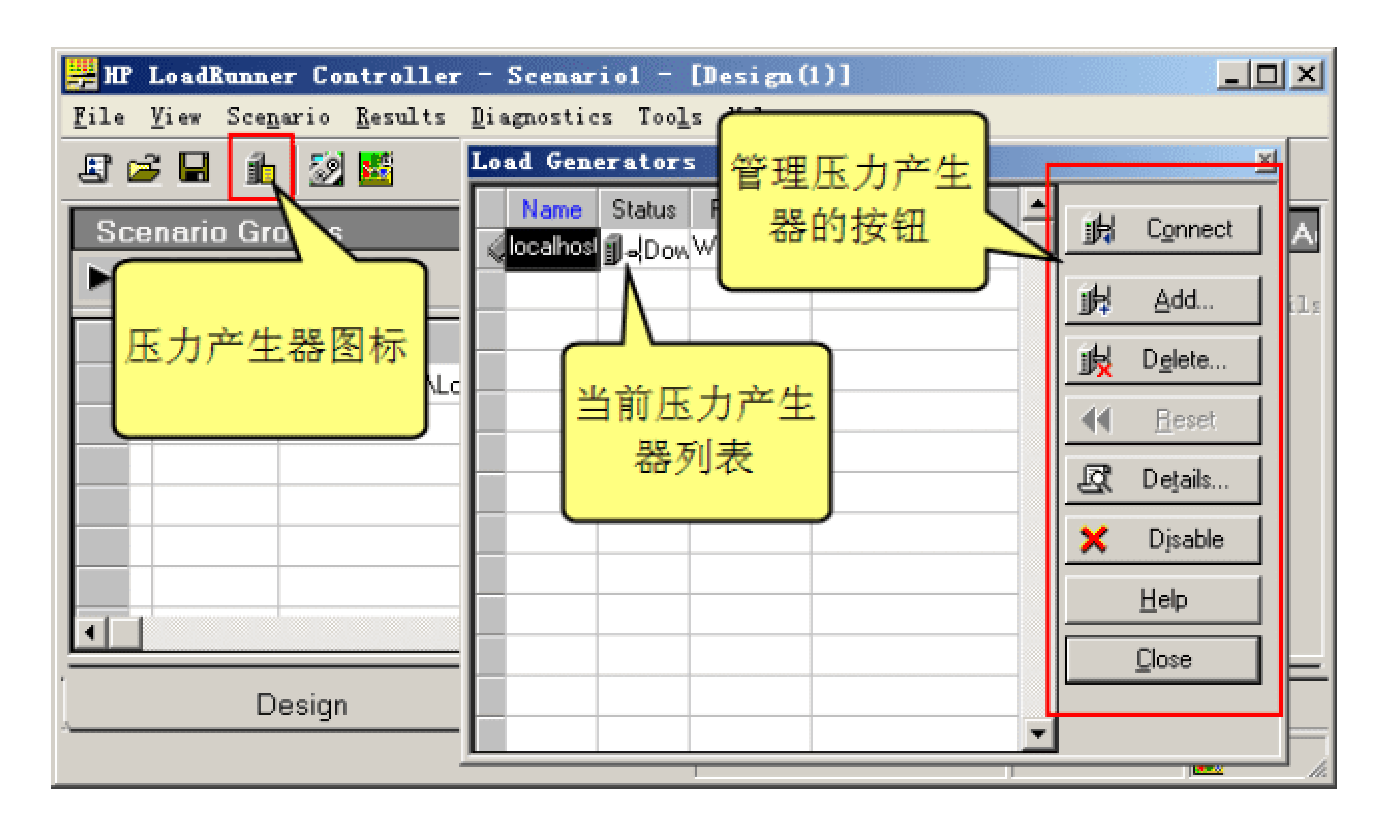


图 10-13　打开压力产生器及其显示界面

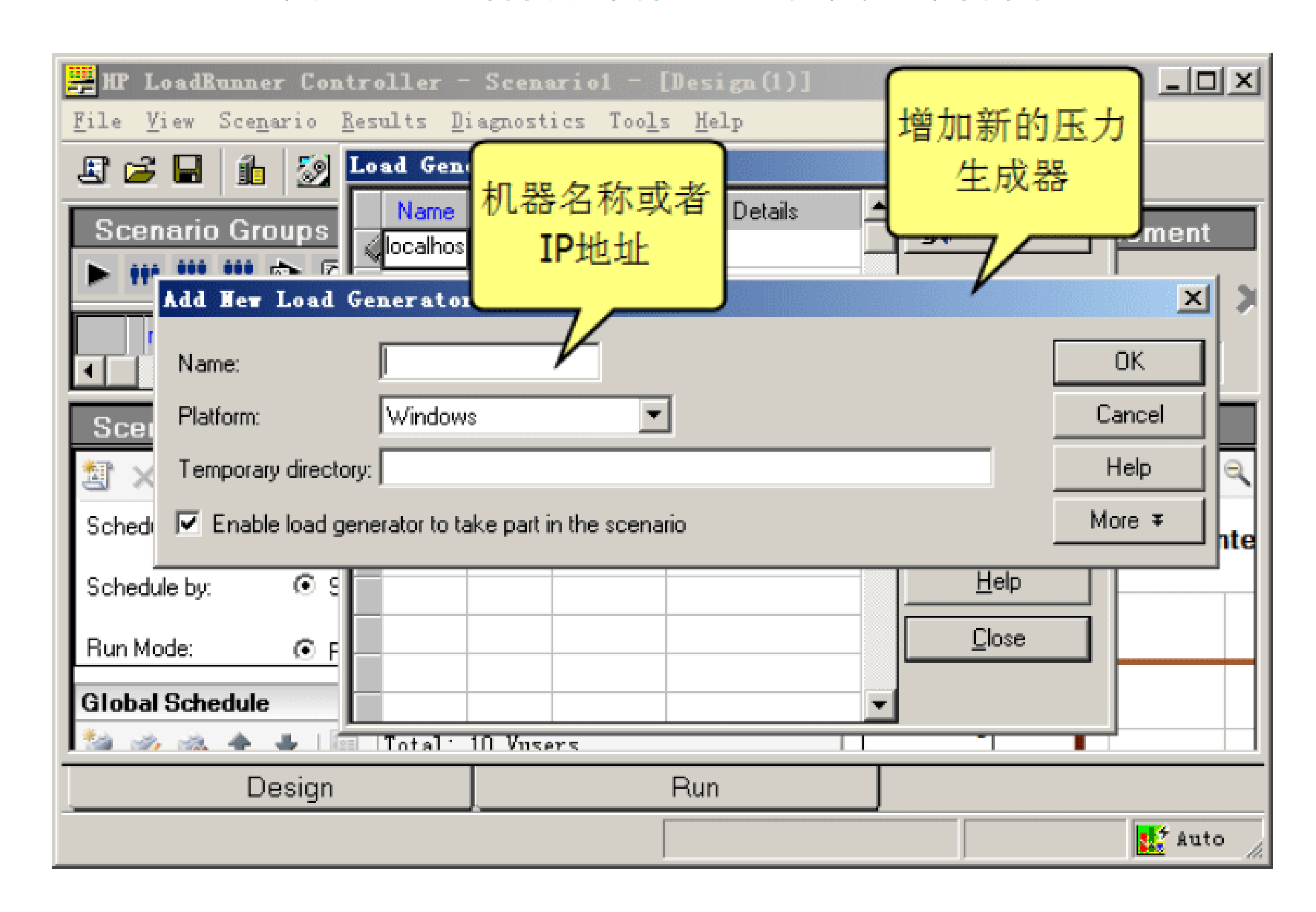


图 10-14　增加压力生成器

2．压力产生器的更多设置

以上只是对于压力生成器的简单设置，如果对它有更多的要求，则可以单击 More（更多）按钮，打开界面如图 10-15 所示。可以看到，在图 10-15 中可以对防火墙、SSL、场景运行时文件保存位置、Unix 系统登录名及 Shell 种类、终端服务设置等多项数值进行修改。由于这些设置涉及很多外部测试环境的知识，与实际情况结合也较紧密，感兴趣的读者可以在工作中逐渐领悟并掌握它们的用途。

单击 OK 按钮，这个压力生成器就增加成功了。依次类推，还可以增加更多的压力生成器。如图 10-16 所示为多个压力生成器的列表。

在图 10-16 中，有两个压力生成器，一个名为 localhost，代表本机，另一个名为 Vista-User，代表另一台 Vista 机器。我们也可以采用类似 Excel 的操作方式，直接在已有记录下面的新行中输入文字来创建新的压力生成器。

【压力生成器的状态与连接】

在图 10-16 中可以注意到每一个压力生成器都有状态栏，分别是 Ready（就绪）和 Down（关机）。单击 Vista-User 这一行以选中该压力生成器（即名称为 Vista-User 的电脑），再

单击右边的 Connect（连接）按钮，如果成功，则该行的状态会变为 Ready（就绪），否则状态变为红色字体的 Failed（失败），并在该行的详细信息一栏中显示原因，如图 10-17 所示。在运行场景之前，测试工程师需要保证列表中的压力生成器可以成功连接，即均为就绪状态。

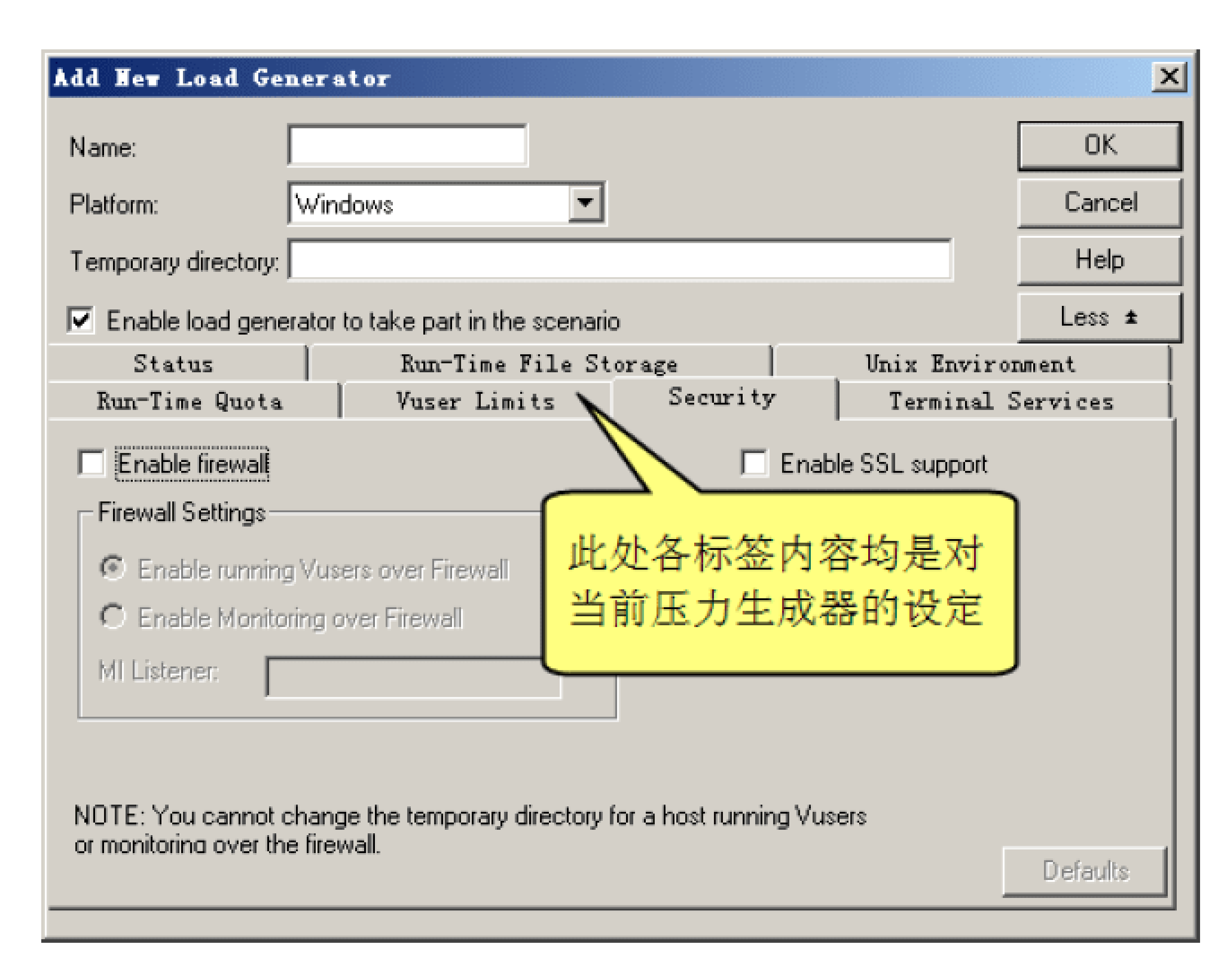


图 10-15　压力生成器的更多设置

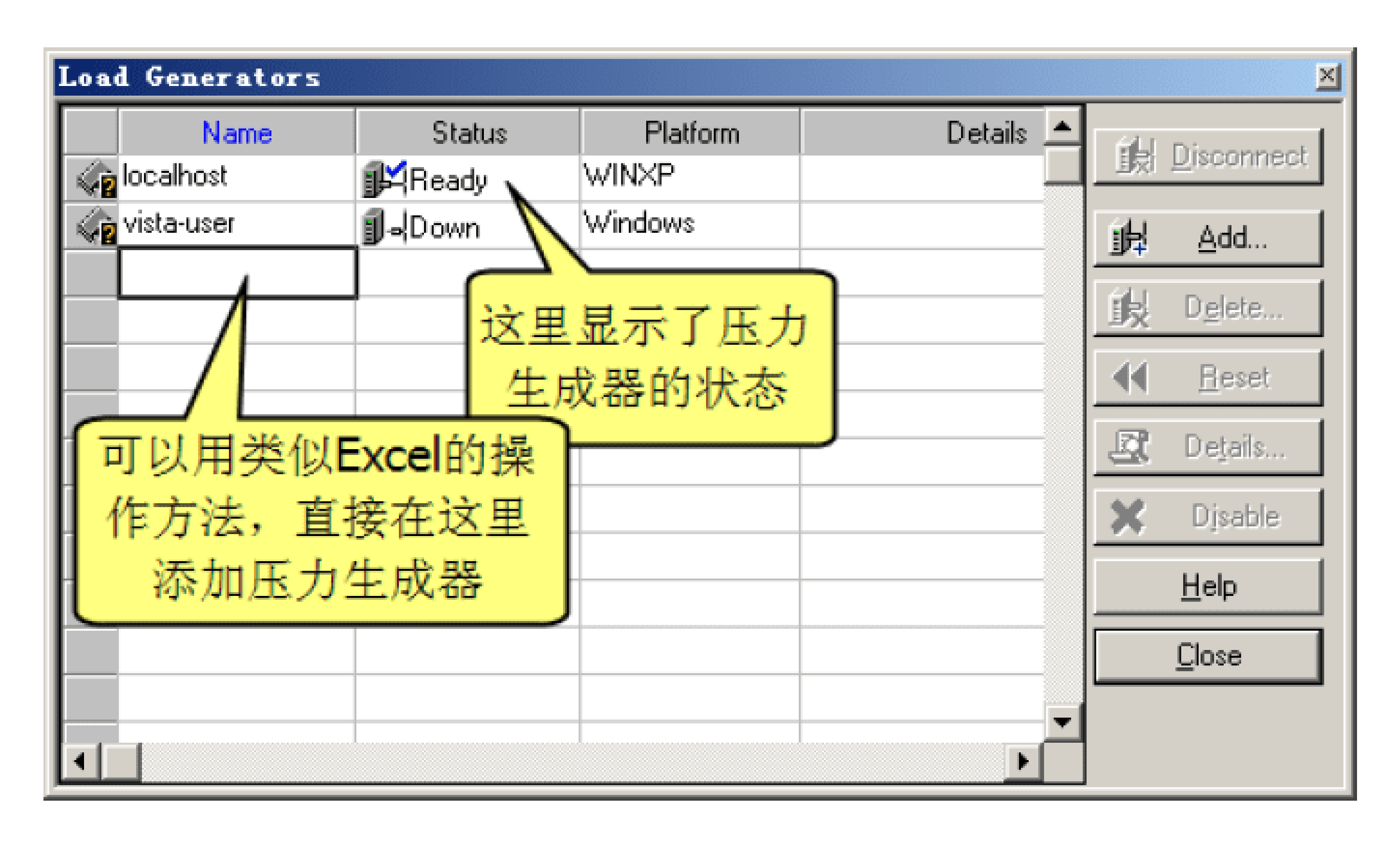


图 10-16　压力生成器列表



图 10-17　压力生成器的状态

【用户组与压力生成器】

用户组和压力生成器也可以有机地结合起来。比如不同的用户组采取不同的压力生成器：为系统为 WindowsXP、Vista 的计算机分别创建压力生成器，与不同的用户组联系起来，从而获得不同系统下的数据。

10.1.7　用户组的增加与修改删除

在对手动场景相关的大致界面和操作熟悉之后，我们可以着手创建场景中的用户组。用户组的分类方法则需要依据测试工程师对于真实情况的理解而定。

在我们一直引用的在线订票场景中，用户组可以有如下几种分类方法：

- 根据浏览器来分类，比如 IE 6、IE 7、Firefox、Safari 等各分为一组。
- 根据操作系统来分类，比如 Windows、Linux 等各分为一组。
- 根据用户操作熟练程度来分类，比如初次用户（两次操作之间间隔较长、思考时间较多）和熟练用户、高速接入网络用户和低速接入网络用户等各分为一组。
- 根据脚本来分类，比如订票与付款的各分为一组。

还有很多种方法，测试工程师可以依据实际情况进行选择。

增加、修改和删除用户组很简单，在图 10-9 中通过单击相应增加和移除按钮即可。这里以增加用户组为例讲解实际操作。

如图 10-18 显示了单击增加用户组后弹出设置窗口的界面，可以修改 Vuser Quantity（虚拟用户数量）微调框以确定新组包含多少虚拟用户。

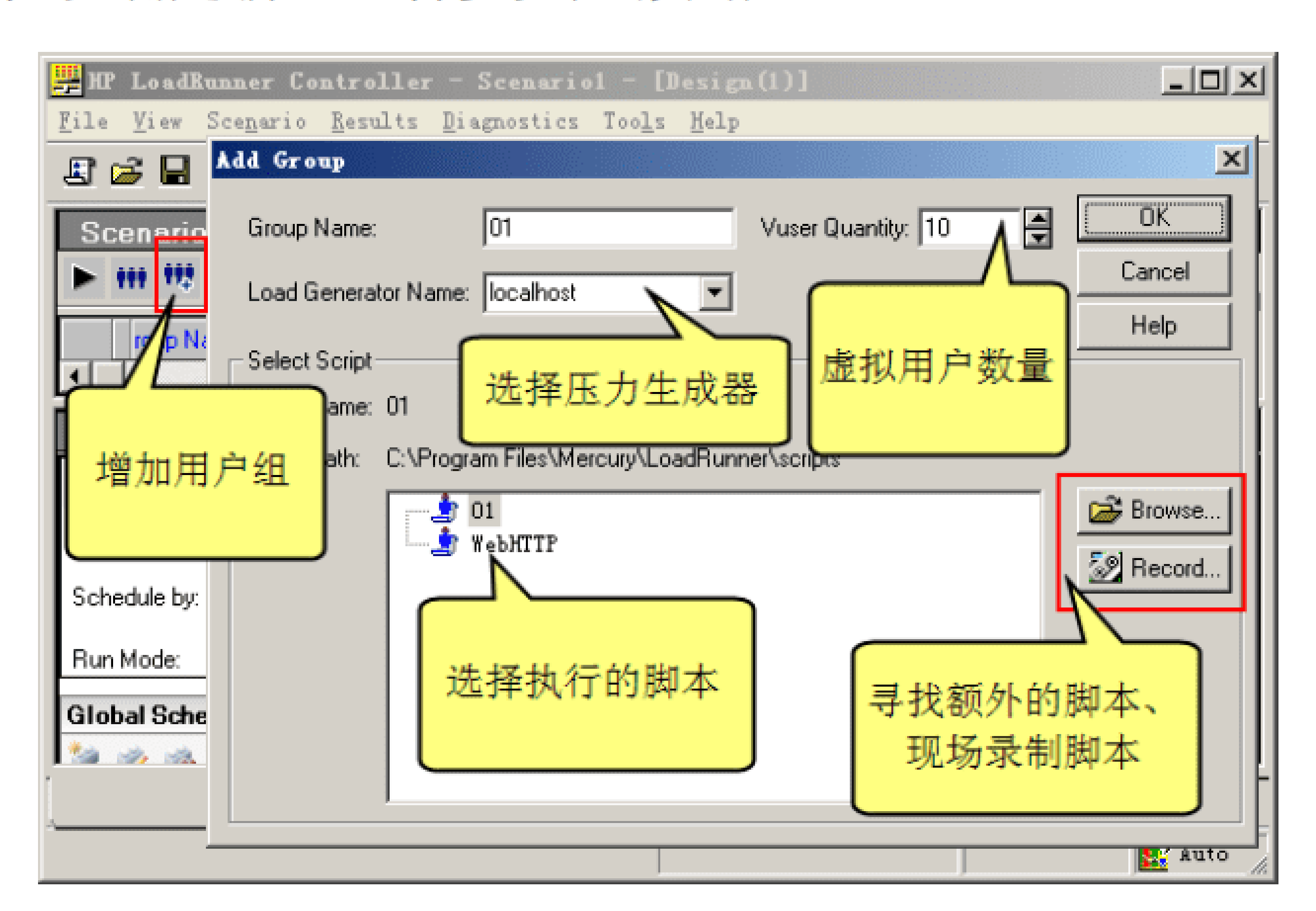


图 10-18　增加用户组及用户组设置

图 10-18 中各个选项就将之前的诸多设置联系了起来。组名字用来标识这一类用户的特征，默认为执行脚本的名称，可以根据实际需要修改为更简明易懂的文字。压力生成器下拉列表框可以令组内的虚拟用户运行在那台电脑之上。虚拟用户的数量决定了组的大小和产生负荷的大小。执行的脚本则规定了当前用户组的操作，如果无法满足需要还可以通过右边的两个按钮添加或者录制。因此，有了软件（执行的脚本）、硬件（压力生成器）和数量（虚拟用户数量）3 者的配合，一个能够代表真实世界、却由虚拟用户组成的用户组就创建完毕了。

在创建好用户组之后，单击“确定”按钮，此时控制器中场景用户组的界面将类似图

10-19 所示。

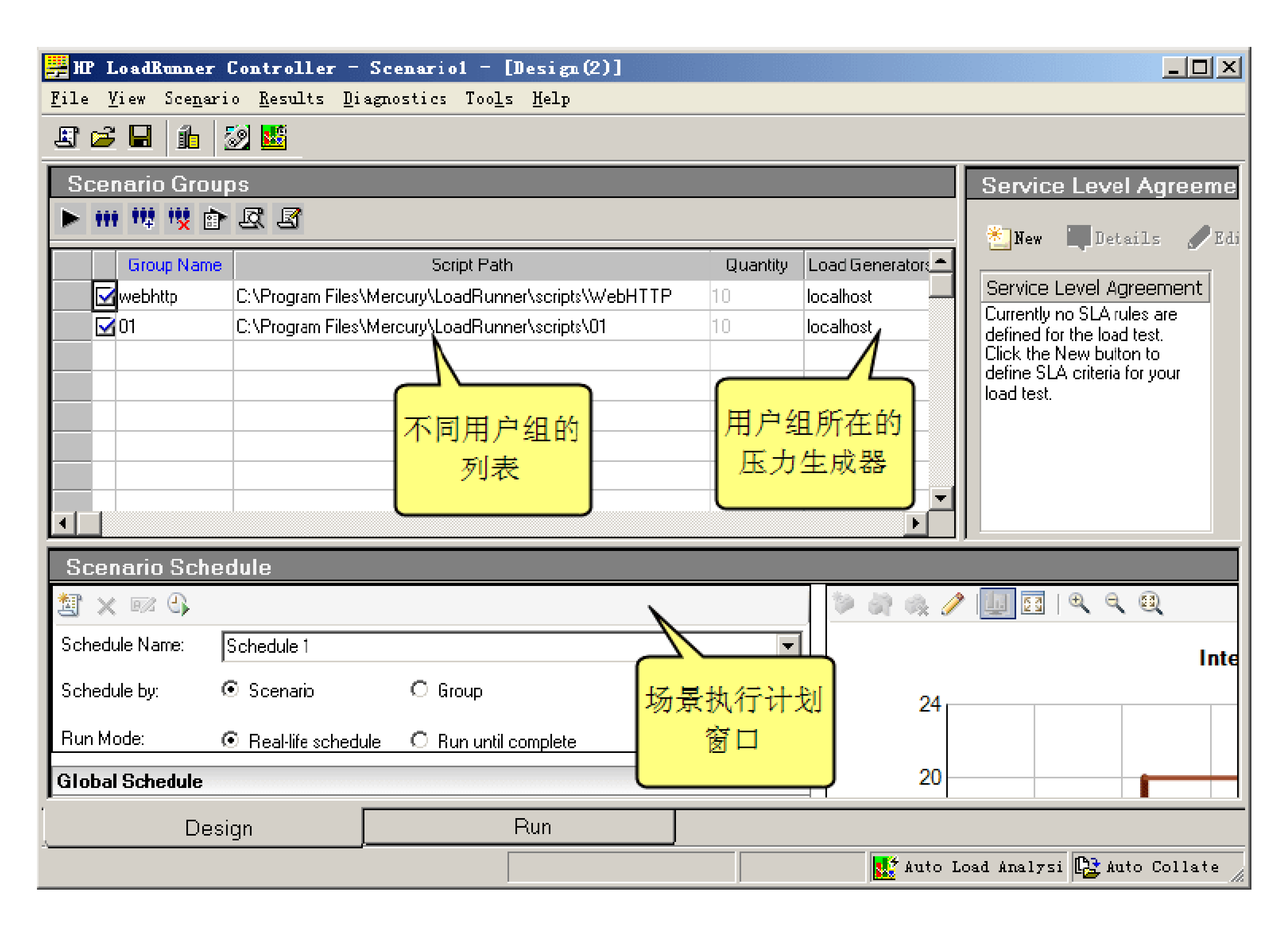


图 10-19　用户组添加完毕后控制器的界面

至此，用户组的添加已经大功告成。在下面将介绍场景用户组导航条中其他的几个功能按钮。

10.1.8　运行时设置（RTS）

在图 10-10 所示的场景用户组管理工具栏中，还有一个 Run Time Setting（运行时设置）图标按钮。单击后弹出运行时设置对话框，如图 10-20 所示。可以发现，这个窗体实际上和前面章节中介绍的脚本运行时设置是一样的，因此这里就不再赘述了。

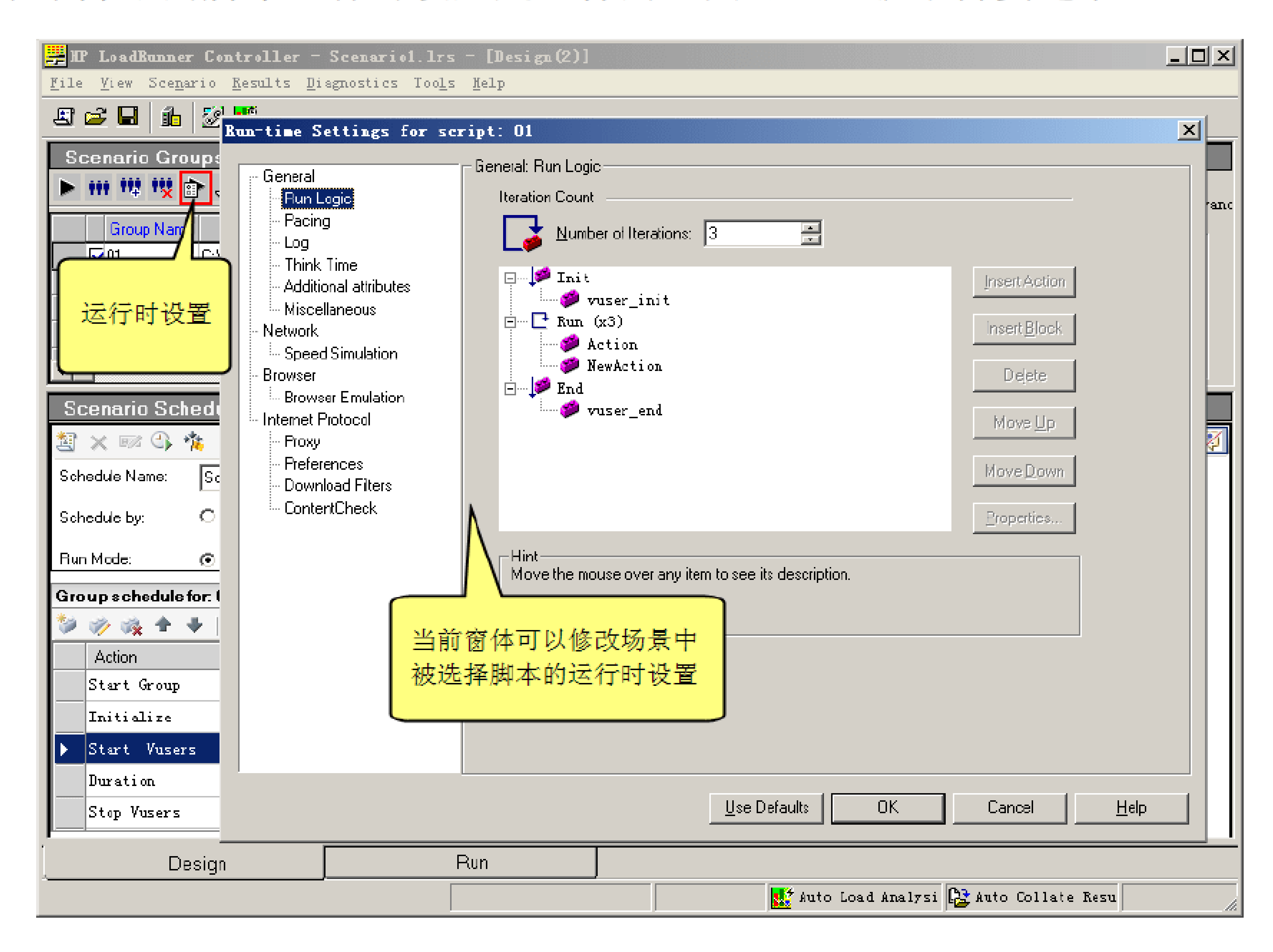


图 10-20　场景用户组管理中对脚本进行运行时设置

10.1.9　场景详细信息设置（Details 按钮）

在场景用户组管理工具栏中，最后还有两个图标，一个是显示场景详细信息设置的 Details（详细信息）按钮，另一个在单击后，则会打开 VuGen 程序，显示当前用户组执行脚本的内容。实际上，单击详细信息设置按钮后弹出窗体的界面也包含了显示脚本按钮的功能，如图 10-21 所示。

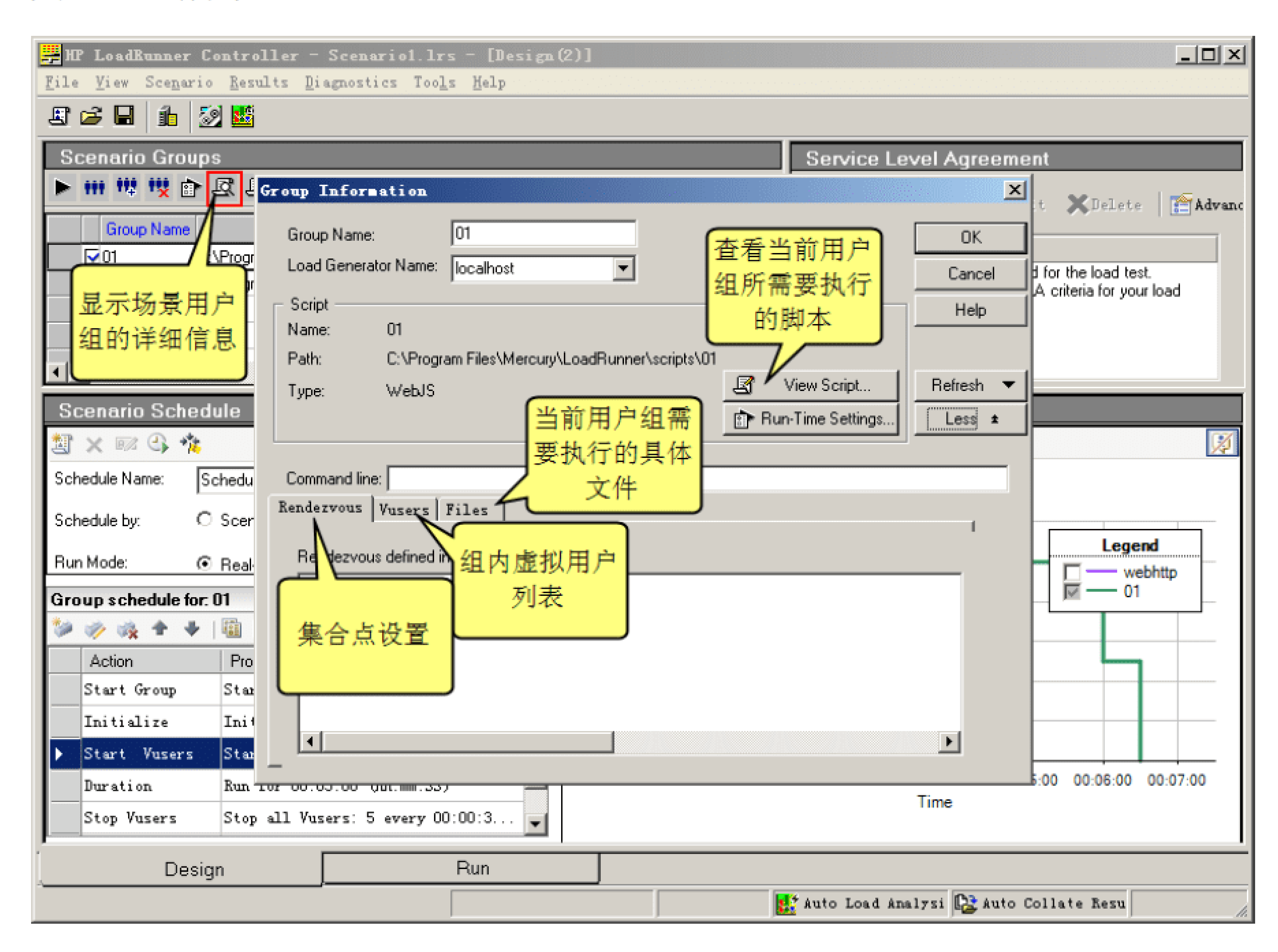


图 10-21　显示场景用户的详细信息

图 10-21 的用户组详细信息显示窗体完整地列出了当前用户组的各种设置。测试工程师可以查看需要执行的脚本、脚本中具体的文件、脚本运行的设置、虚拟用户的列表以及集合点的设置等。在平时的工作中，无论是在自己创建用户组阶段还是浏览同事已经建好的用户组信息都是很有裨益的。

在这里，我们接触到一个新的名词，集合点（Rendezvous）。10.2 节将重点讲述集合点的使用。

10.2　集　合　点

集合点 Rendezvous 这个英文单词来自于法语，意思是军队集合的意思。在 LoadRunner 中，这是一种形象的比喻：各个虚拟用户在同一时刻完成相同的操作，不就很类似于军队的集合吗？

引入集合点的目的是为了评估真实情况下，多个用户同时进行被测试软件的某个操作时，系统的性能表现。一句话，集合点与系统的并发密切相关。

【集合点产生的背景】

在 LoadRunner 执行场景的时候，每个用户组内各用户执行的脚本虽然都一致，但它们的执行速度却不尽相同（想象真实情况下对某一个固定网站的每一次访问，响应时间都不是完全一样的），这也正是性能测试需要采集的数据之一。因此，有的虚拟用户可能很快就执行完全部脚本，而有的则需要更长的时间。为了能够测试并发性能，确保在某 1 时刻，确实有指定数量的虚拟用户在进行同样的操作，LoadRunner 采用了集合点的技术。

10.2.1　集合点的设置步骤

在具备了集合点的基本知识之后，让我们开始应用在实际工作中。集合点的设置分为两个步骤：

（1）在脚本中需要测试并发性能的操作之前加入集合点。

（2）在场景中设置集合点的执行规则。

由此可见，集合点需要脚本录制后的修改与场景设置的配合，因此，本书并没有在脚本录制章节中介绍，而是将这部分内容放在了本章。

10.2.2　在脚本中加入集合点

单击场景用户组导航条中 View Script（查看脚本）按钮，控制器将自动打开 VuGen，并显示当前用户组需要执行的脚本内容。在脚本的 Action 部分，将光标放置于需要设置集合点的前面，选择 Insert|Rendezvous（插入|集合点）命令，如图 10-22 所示。

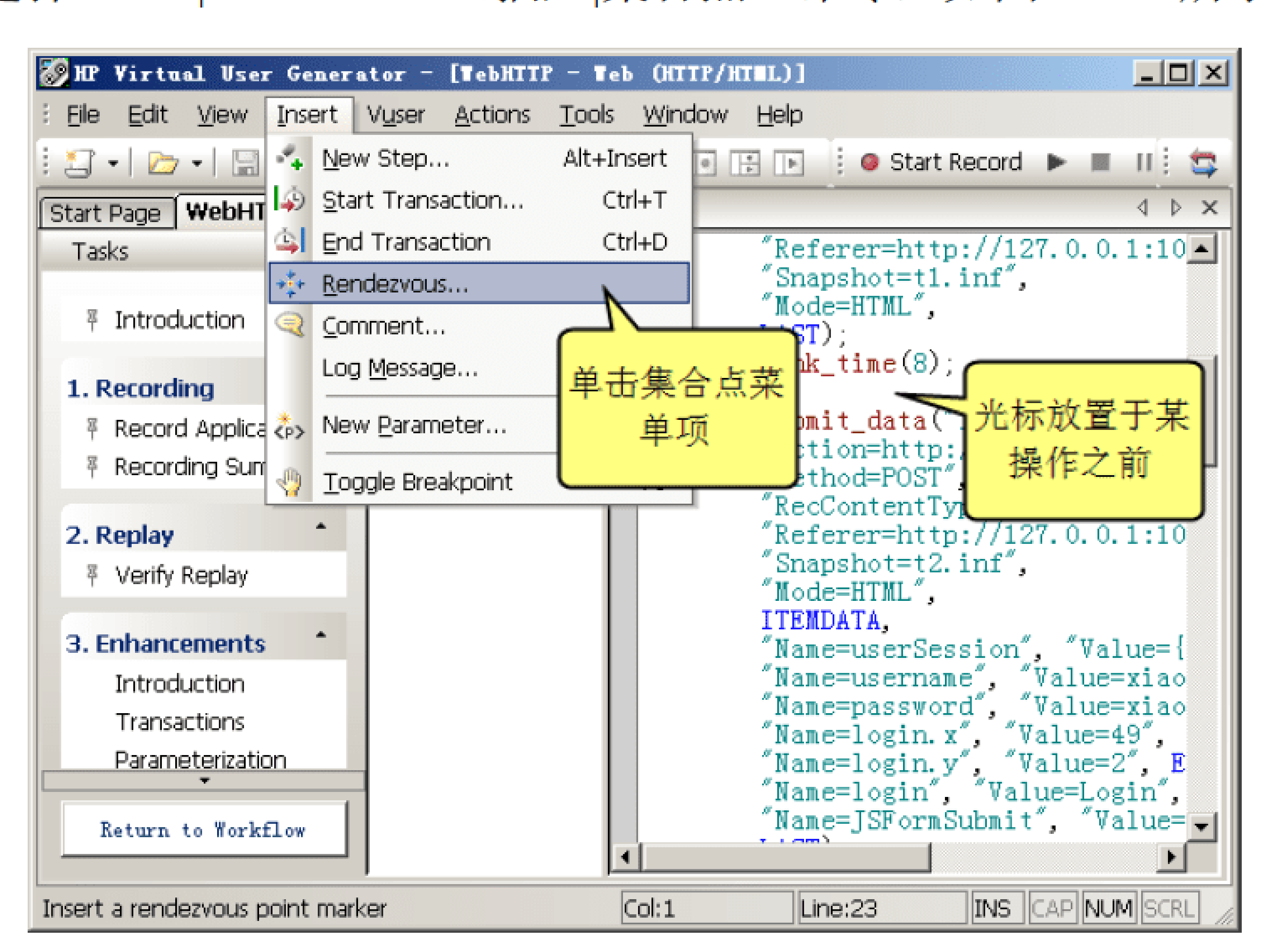


图 10-22　在脚本中插入集合点

之后插入集合点的界面，如图 10-23 所示。实际上，集合点起到了一个类似集结号的作用。当 LoadRunner 的某一个虚拟用户执行脚本的时候，一旦发现当前行是集合点，就会被控制器命令停止前进。控制器同时清点到达此地的虚拟用户人数，如果达到了场景设置中的要求，再进行下一步的操作，这样就保证了下一步操作具备指定数量并发的特性，从

而可以获得准确的并发数据。

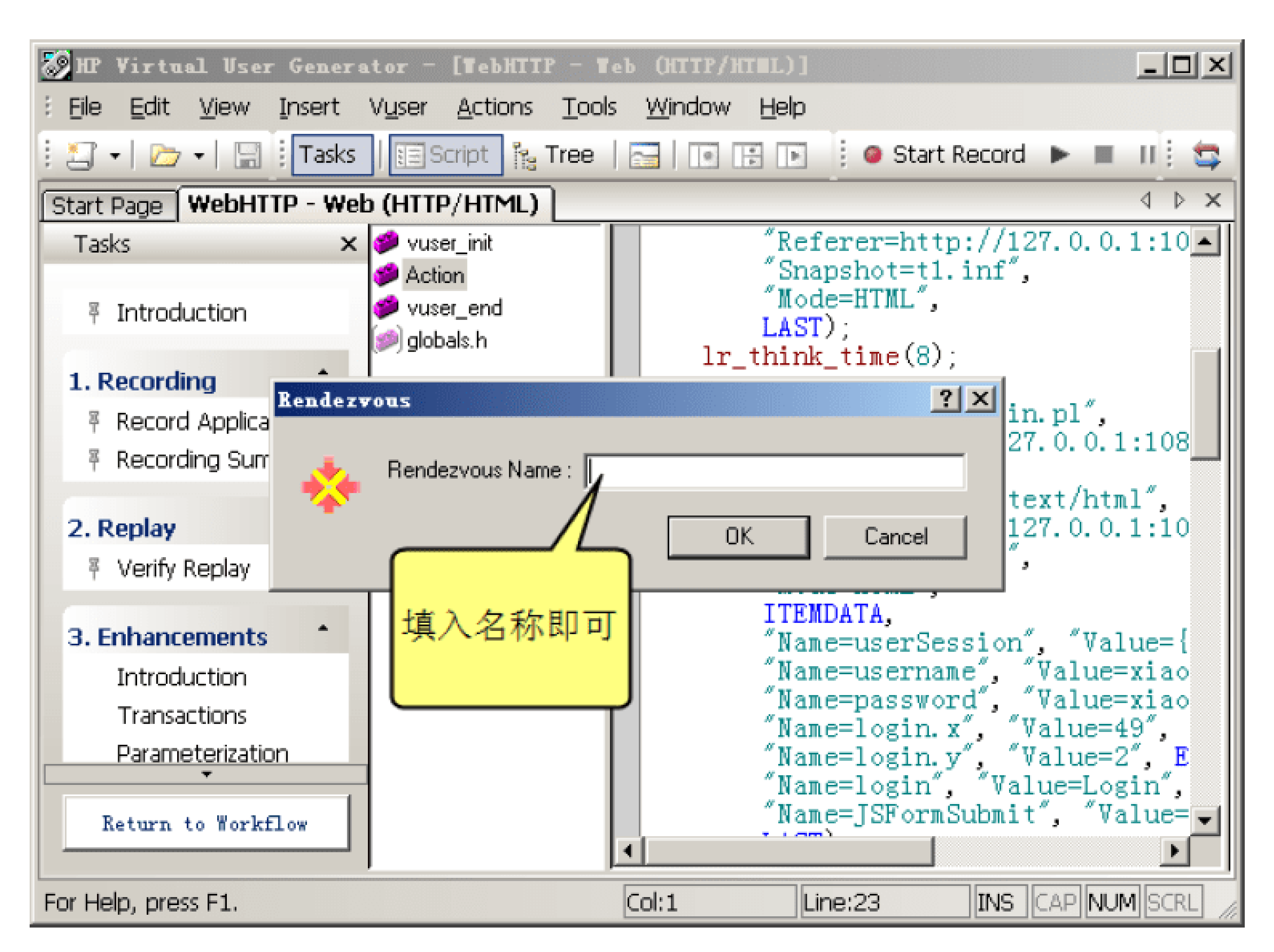


图 10-23　在脚本中增加集合点

在图 10-23 中，测试工程师只需要在 Rendezvous Name（集合点名称）文本框中输入一个易懂、好记忆的名字即可，在这里举例添加了 Concurrent-Login 这个名称。单击 OK 按钮添加集合点成功，可以发现，脚本中增加了一行，在图 10-24 中用方框标出。

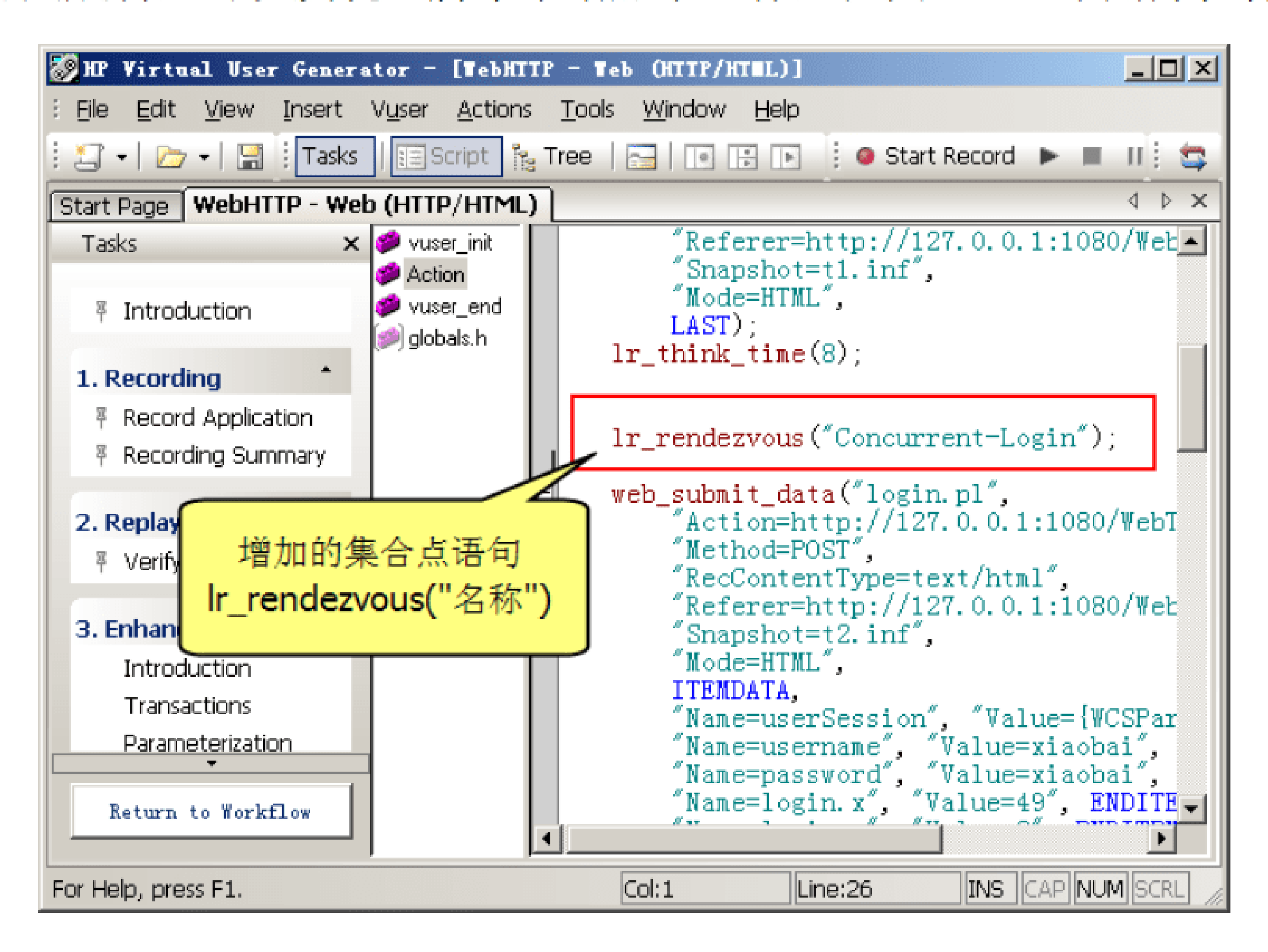


图 10-24　增加集合点后脚本的变化

测试工程师可以根据实际需要，在脚本的不同操作之前都增加集合点。

10.2.3　在场景中配置集合点

在 10.2.2 节我们在脚本中增加了集合点，但是，并没有设置在这个集合点中需要集结

多少个虚拟用户，该设置是在场景设置中完成的。

回到控制器的界面，单击场景用户组中的 Details（详细信息）图标按钮，在其中的界面中单击 More（更多）按钮，可以发现，集合点 Concurrent-Login 已经显示在列表当中，如图 10-25 所示。在个别的情况下，可能需要单击 Refresh（刷新）按钮来强制信息更新才能看到新加入的集合点。

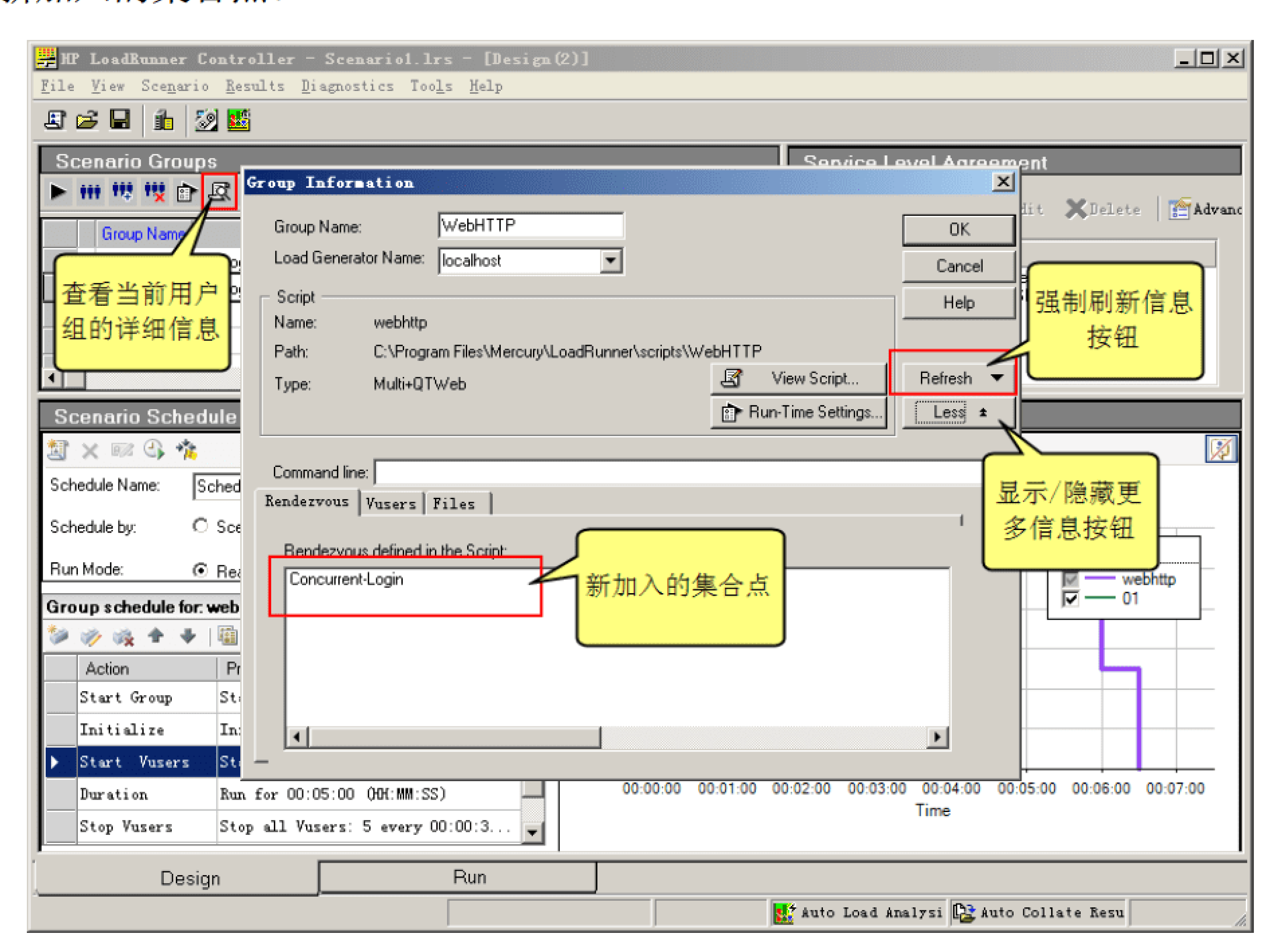


图 10-25　新加入的集合点已经在当前用户组的详细信息中显示出来

集合点能够真正发挥作用，还需要规定在这个集合点上要集结多少个虚拟用户。这个设置是通过选择控制器 Scenario（场景）菜单下 Rendezvous（集合点）子菜单项来实现如图 10-26 所示的窗体。在该图中有一个 Policy（规则）按钮，可以用来设置在集合点处解散的规则。

【解散规则】

所谓解散规则，就是何时开始执行集合点后脚本语句的规则。注意，集合点后的那条语句正是需要测试并发的操作。

LoadRunner 提供了 3 种解散规则，分别如下：

- ❑ 当所有正在执行脚本的虚拟用户都到达集合点时，可以执行下一步操作。这是 LoadRunner 默认的集合点解散方式。
- ❑ 当固定百分比的虚拟用户到达集合点时，可以执行下一步操作。
- ❑ 当固定数量的虚拟用户到达集合点时，可以执行下一步操作。

同时，LoadRunner 还可以设置虚拟用户之间互相等待的超时时间，默认是 30 秒。这个数值的意义在于，当部分虚拟用户已经执行到集合点，而其他一些用户尚未到达，集合点处停留的用户等待下一个到达集合点用户的最大时间。超过这个时间，就不再等待了。

这几种规则都可以在图 10-26 中单击 Policy（规则）按钮后弹出的窗体中设置，如图 10-27 所示。

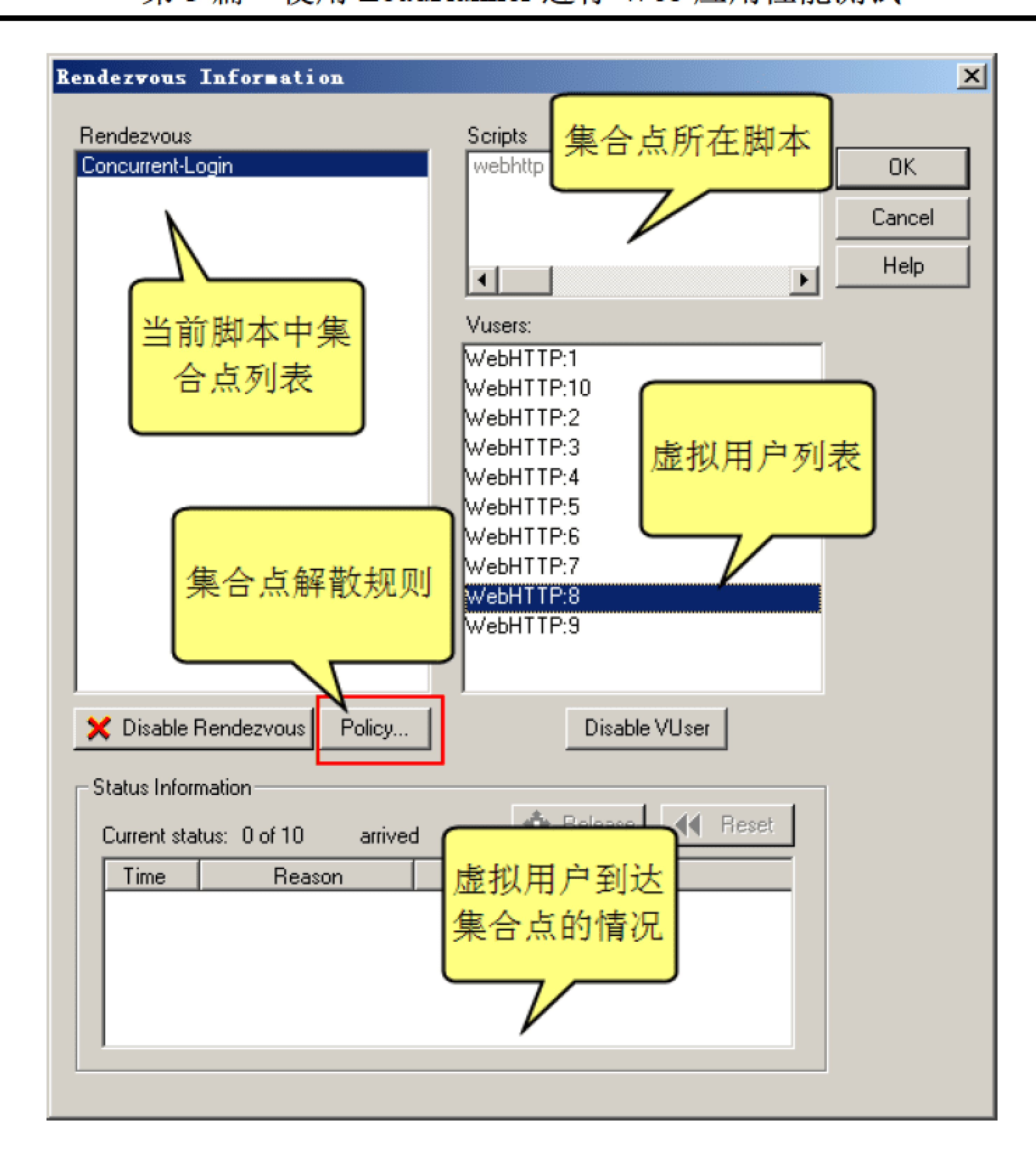


图 10-26　显示与设置当前脚本中集合点的具体信息

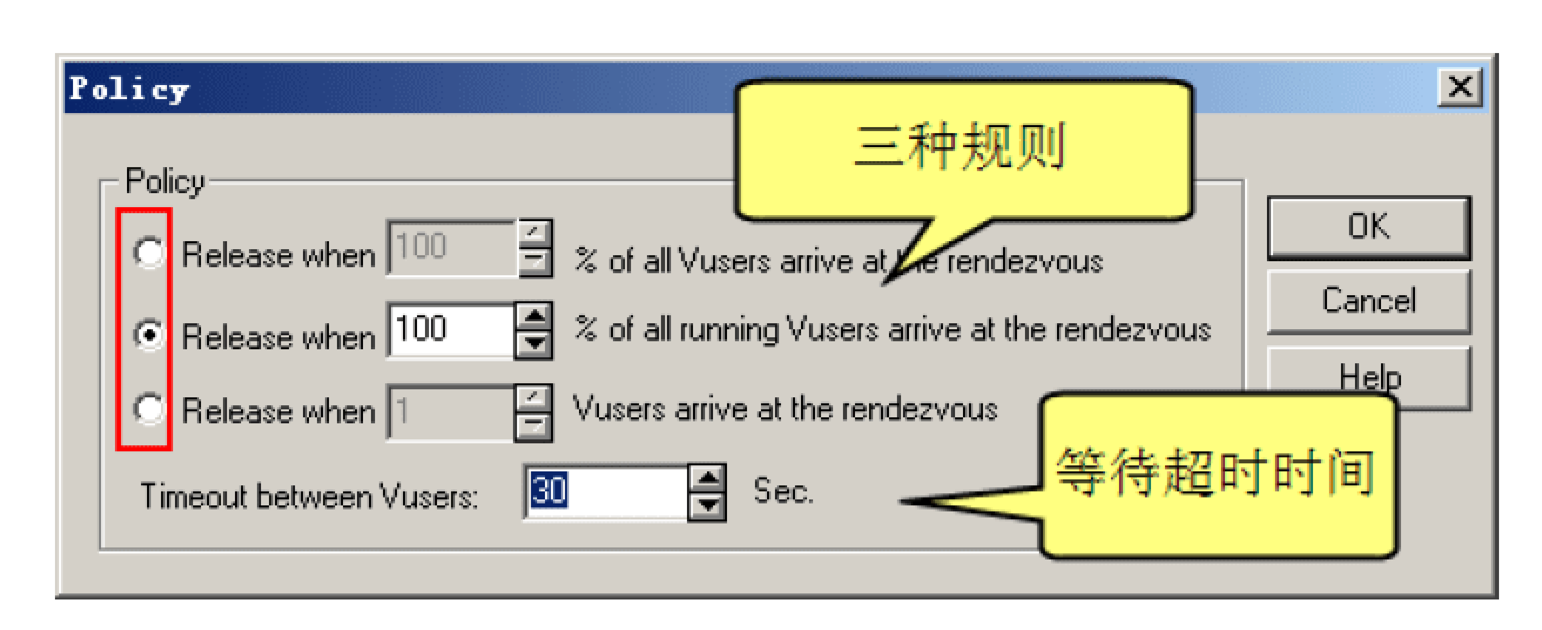


图 10-27　设置集合点解散规则

在图 10-26 中还有几个按钮用来禁止集合点和具体某个虚拟用户，因为用法简单，就不再介绍了。图 10-26 的下方的 Status Information（状态信息）列表框还可以显示了当前集合点虚拟用户的集结情况，方便在运行时查看。

截至目前，我们已经把场景中的两个重要因素：虚拟用户所组成的用户组和机器组成的压力生成器介绍的差不多了。这里用下面的几句话稍微做个总结：

（1）组成用户组有两个因素：脚本与虚拟用户。脚本规定了虚拟用户在测试中执行的内容（各个操作）和方式（思考时间、集合点）。执行相同操作的一类虚拟用户组成了用户组。

（2）不同的机器可以成为不同压力生成器，在执行场景的时候要确保控制器可以与它们连接。

但是，只有用户组中的“人”和压力生成器中的机器还不够，测试工程师还需要在合适的时间把这两个团队发动起来，于是，规定场景何时及怎样运行的执行计划应运而生。

LoadRunner 提供了很好的计划功能，在前面的图 10-19 左下方，就可以发现场景计划（Scenario Schedule）窗口，这方面的具体内容将在 10.3 节进行讲解。

10.3　场景的执行计划

从图 10-19 中可以看出，场景计划窗口分为明显的两部分，左边是计划设置和属性窗口，右边则是一个场景中虚拟用户随时间的变化走势图，非常直观。在本节我们将学习如何通过这两个设置窗口对场景的执行做出计划。

10.3.1　熟悉设置场景运行计划界面

设置场景计划的属性窗口分为上下两个区域，如图 10-28 所示。它们用来分别设置场景执行计划的总体规则和场景执行过程中各操作（Scenario Action）这两个部分。所谓场景操作，包括初始化用户组、启动用户组各用户以及停止虚拟用户的全过程，依据设置不同，执行过程中可以最多有 5 类操作，分别是：启动用户组（Start Group）、初始化（initialize）、启动虚拟用户（Start Vuser）、持续运行（Duration）和停止虚拟用户（Stop Vusers）。测试工程师可以对这几种操作进行修改的是虚拟用户的数量，以及用户从 0 增长到指定数值所经历的变化规律。

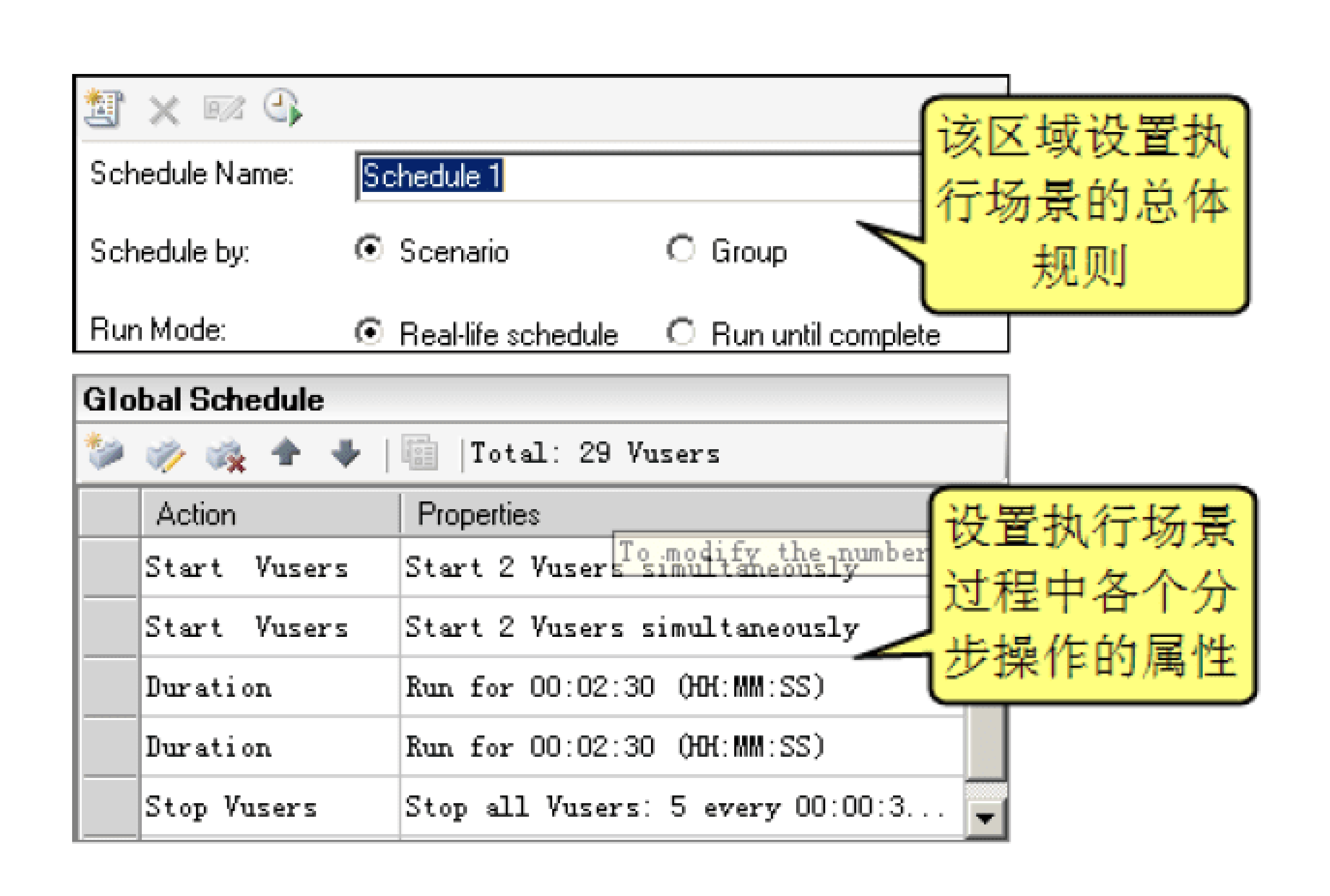


图 10-28　设置场景计划的属性窗口

可以注意到，执行计划的名称可以通过修改图 10-28 中 Schedule Name（执行计划名称）下拉列表框内文字来实现。

10.3.2　设置场景开始运行的时间

在图 10-28 的上半部分，有一个多图标按钮组成的工具条，其中单击各个图标的功能如图 10-29 所示。图 10-29 中的各项功能都比较简单，这里简要讲解一下计划的定时执行设置。单击该图标后系统弹出计划运行时间设置窗体，如图 10-30 所示。

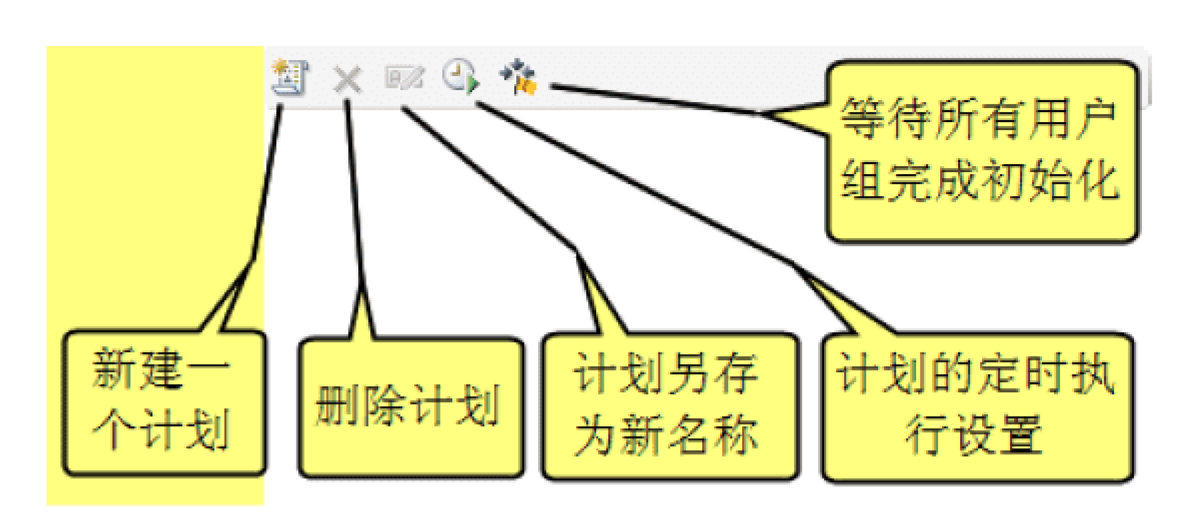


图 10-29　场景计划执行信息设置工具条

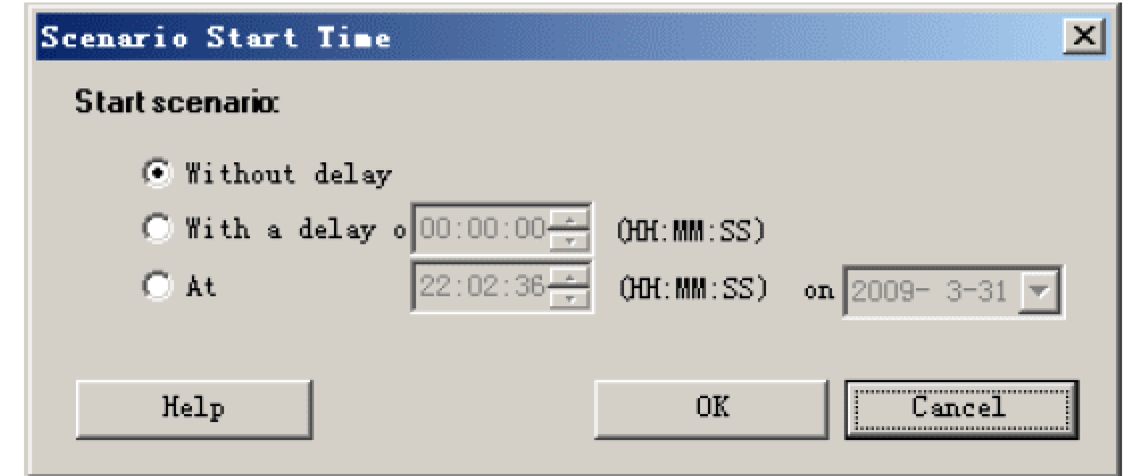


图 10-30　设置计划执行时间

从图中可以发现，场景的运行时间可以是：

- 立即执行（Without delay），这也是默认值。
- 延后执行（With a delay），场景将在设定的时间之后执行。
- 定时执行（at），将在指定时间执行场景。

读者可以根据实际需要进行更改。

另外，等待所有用户组完成初始化按钮只有在 Schedule By 方式（请见下文）选择为用户组（Group）之后才会显示出来。

10.3.3　设置场景执行的方式

场景执行的方式是通过 Schedule by（启动方式）和 Run Mode（运行模式）两个选项进行设置的。

（1）Schedule By 选项可以设置当前计划是按照场景（Scenario）来执行还是按照用户组（Group）来执行，两种方式的区别在于：

- 场景方式中所有的用户组虚拟用户增长的方式一致，用学校活动来比喻，类似全校所有班级参加的团体操比赛。
- 用户组方式中各用户组中的虚拟用户增长方式可以不同，类似全校各班级自报节目的汇演。

（2）Run Mode 选项则设置了计划的执行信息，分为按真实情况计划（Real-Life Schedule）和按脚本设置运行直到结束（Run until complete）两种。二者的区别在于：

- “按真实情况计划”这种方式可以修改持续运行（Duration）与停止虚拟用户（Stop Vusers）这两种在启动虚拟用户之后发生的场景操作属性，它相对第二种执行方式更接近真实情况，故而得名。
- “按脚本设置运行直到结束”这种方式则无法设置用户组启动后的各操作属性数值。脚本运行开始后，用户组的属性就维持不变了。

不过，不管两种运行模式如何，在测试工程师中途不停止的话，用户组需要执行的脚本都是要执行完毕才停止的。

10.3.4　修改场景操作的具体属性

前文提到，场景中的操作分为 5 类：启动用户组（Start Group）、初始化（initialize）、启动虚拟用户（Start Vuser）、持续运行（Duration）和停止虚拟用户（Stop Vusers）。这

几类操作的属性设置都显示在图 10-28 的下半部分。

【运行模式与场景操作的属性设置】

选择不同的运行模式，可以设置的场景操作范围不同，这一点在前文已经提到了。按真实情况计划可以多设置两类操作，并且还可以在计划中增加这两类操作；而按脚本设置运行直到结束则只能设置初始化和启动虚拟用户这两类的属性，并且无法增加场景操作。

通过双击图 10-28 中的某个场景操作即可打开它的属性设置窗体，分别如图 10-31、图 10-32、图 10-33、图 10-34 所示。各窗体具体设置的含义基本相同，一般都是虚拟用户数量增加的方式（一次增加到最大还是依次增加到最大）与时间设置，读者在实际工作中花几分钟就可以熟悉。

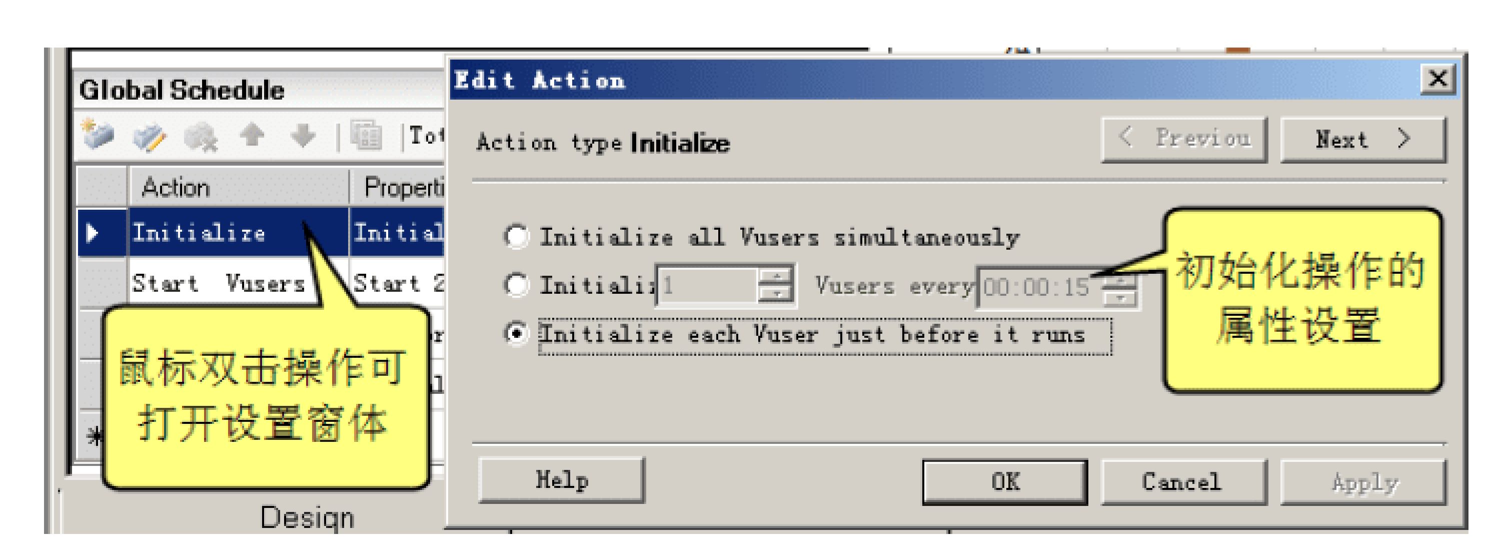


图 10-31　初始化操作的属性设置窗体

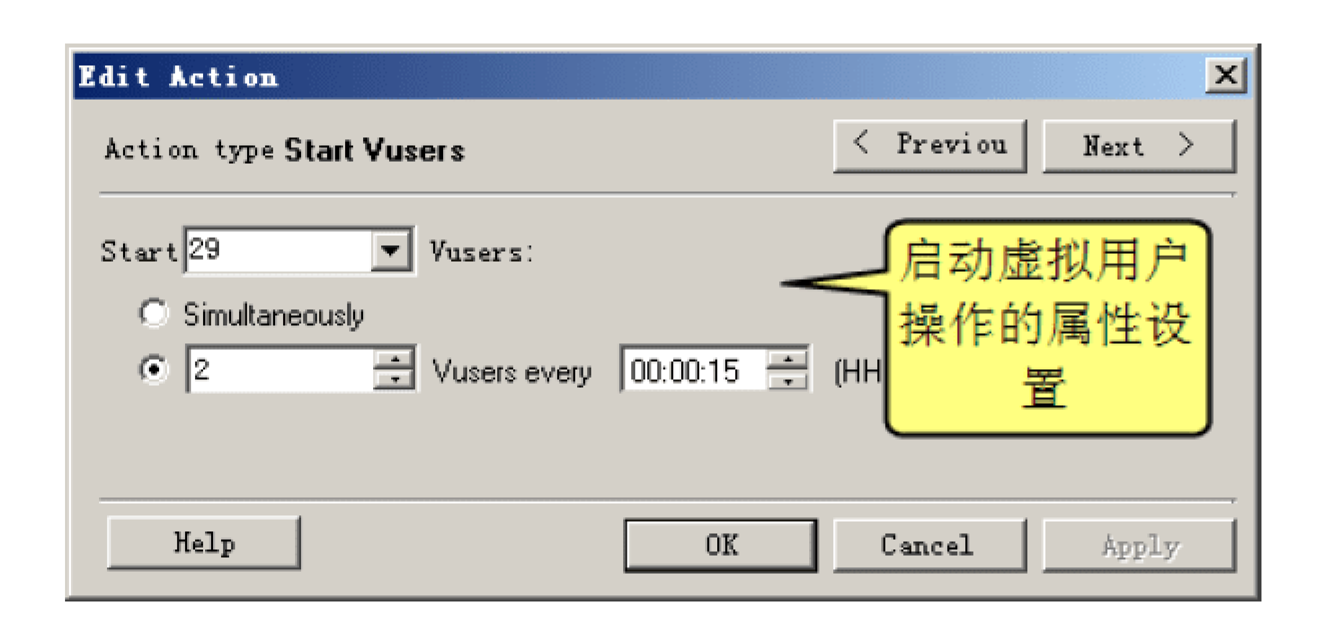
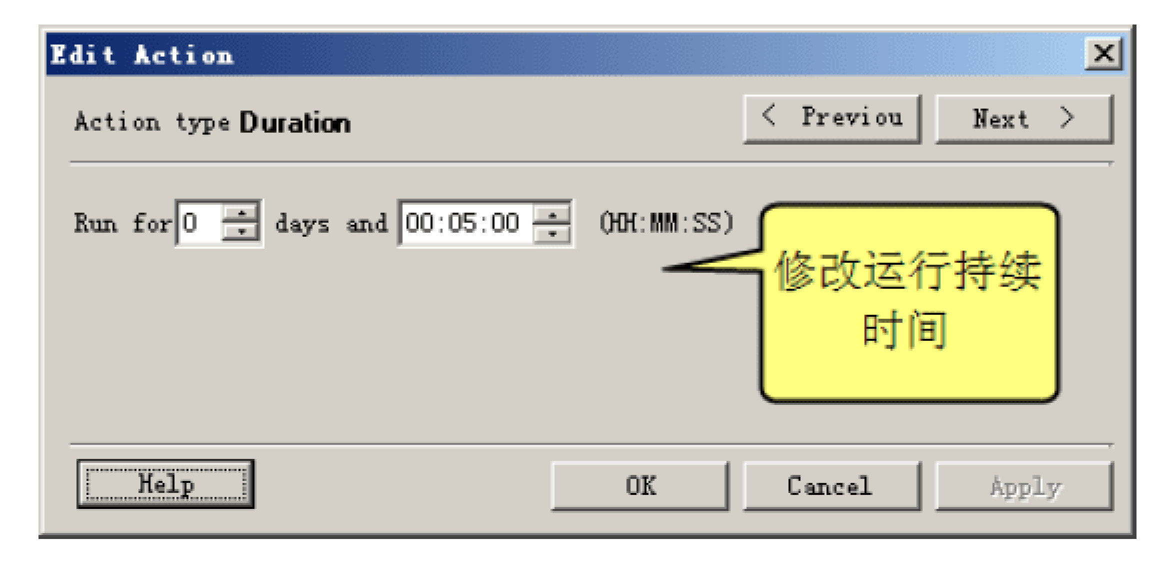


图 10-32　启动虚拟用户操作的属性设置窗体



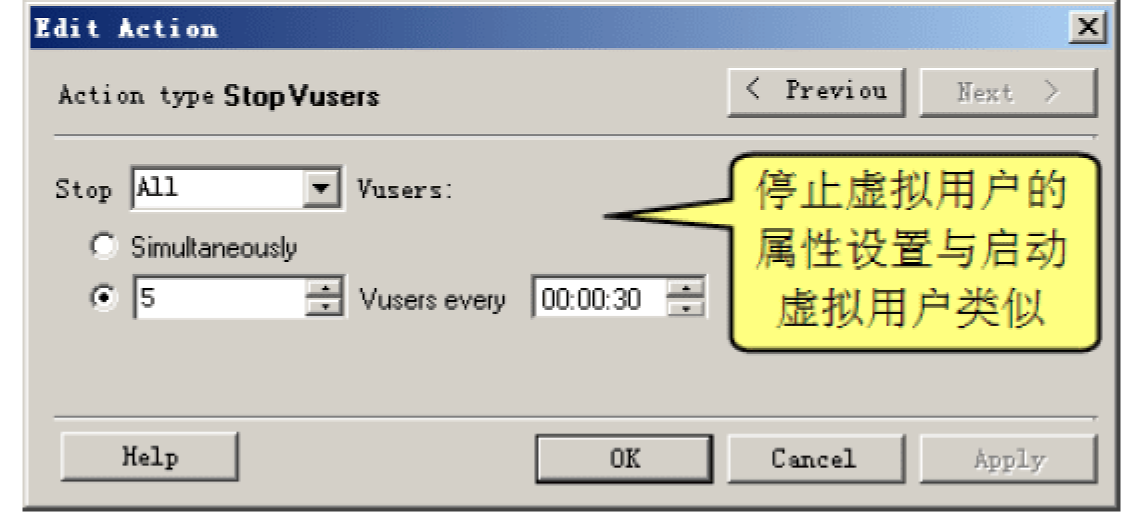


图 10-33　持续运行（Duration）操作的属性设置　　图 10-34　停止虚拟用户操作的属性设置

在选择运行模式为按真实情况计划时，还可以增加某几类场景操作：比如，若打算分批启动虚拟用户，则可以增加一项启动虚拟用户的操作，这样，在场景中就会分两次启动虚拟用户，每次可以启动具体设置的数值。增加场景操作的示意图如图 10-35 所示。

如果读者安装有 LoadRunner 的话，将会发现每次修改一些设置，控制器右下方的图片都可能会发生变化。这是 LoadRunner 很有意思的一项功能：为了直观和理解的方便，系统对当前场景计划做出了一个运行图。实际上，通过对图的修改，测试工程师也可以设置

一些属性，这将是 10.3.5 节的内容。

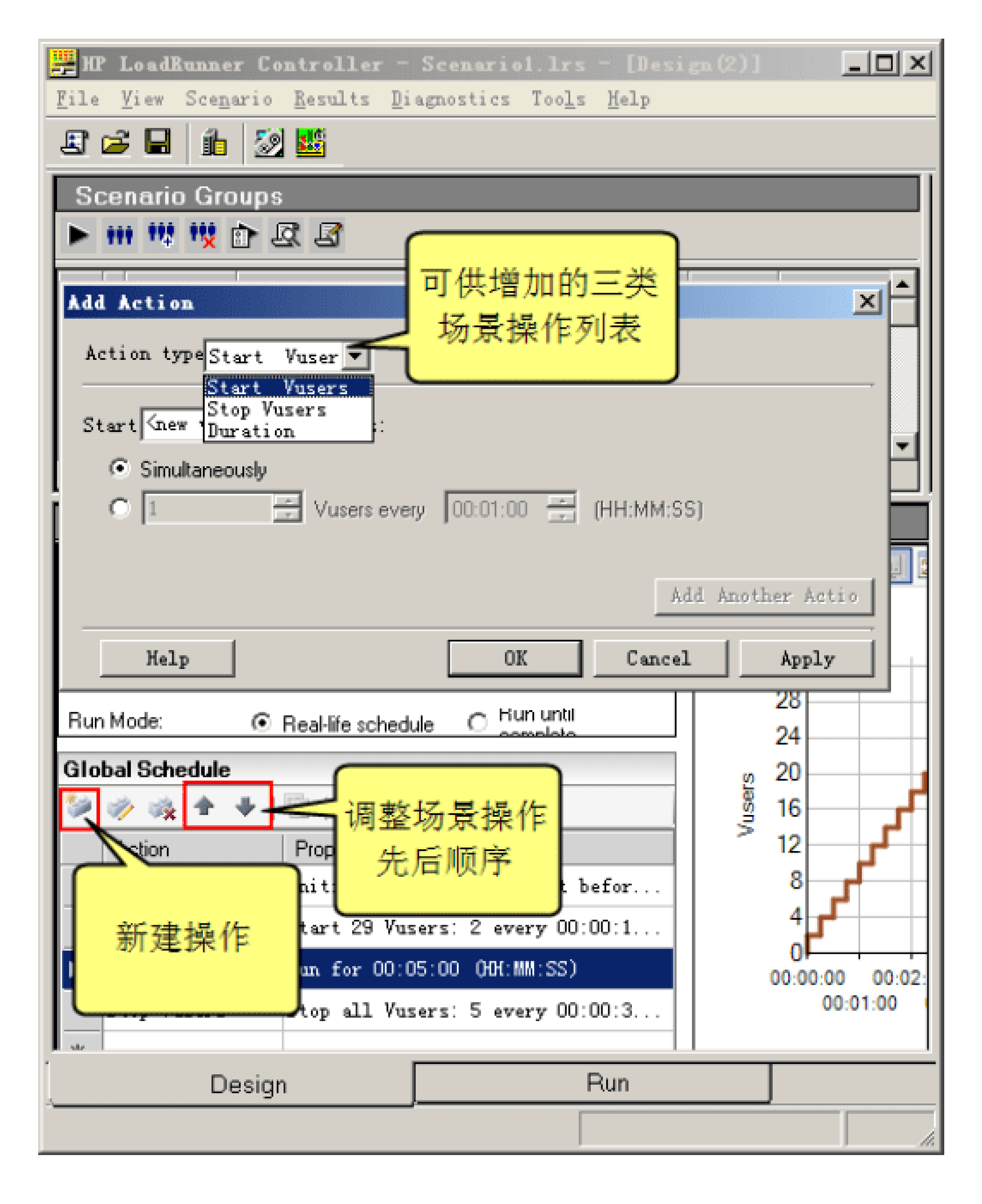


图 10-35　增加场景操作

10.3.5　图形方式设置手动场景的运行计划

图形的优点是非常直观，但前提是人们能够看懂它的图例。如图 10-36 对 LoadRunner 控制器中的运行图做了简单介绍。

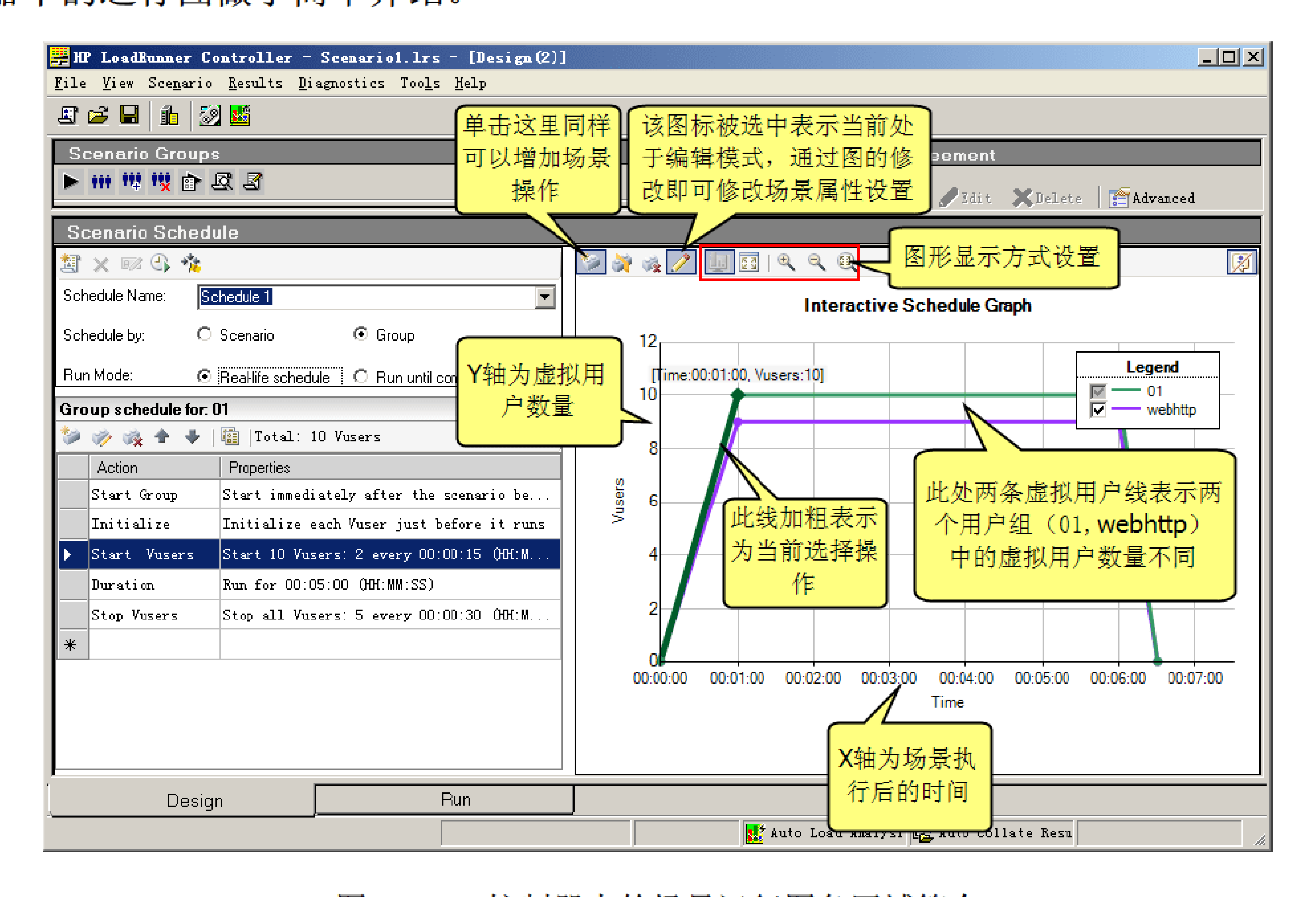


图 10-36　控制器中的场景运行图各区域简介

测试工程师可以通过拖动运行图中各线的连接点（鼠标会变为手型）来修改时间（X 轴数值）和虚拟用户数量（Y 轴数值）。不过，在实际工作中，运行图更多地是用来展示场景运行计划而不是修改属性（可以用 10.3.4 节的方法精确设置各属性值），因此这里不再花费更多的篇幅来讲述。

以上的各种设置和选择都是针对当前场景的，10.4 节要介绍的则是控制器的全局默认设置，它们对于所有的场景，在具体某个场景没有做相应修改的话，都是有效的。

10.4　控制器的全局设置

10.3 节我们详细介绍了创建面向目标的场景与手工场景的各种设置，也提到这些设置都是针对具体的特定测试场景的，如果场景不同或者测试类型不同，数值一般不同。本节所将介绍的控制器全局设置则有些特殊，其中的数值对于该控制器下管理和实现的所有场景都有效。

打开控制器全局设置的方法很简单：选择控制器 Tools|Options（工具|选项）命令即可，如图 10-37 所示。

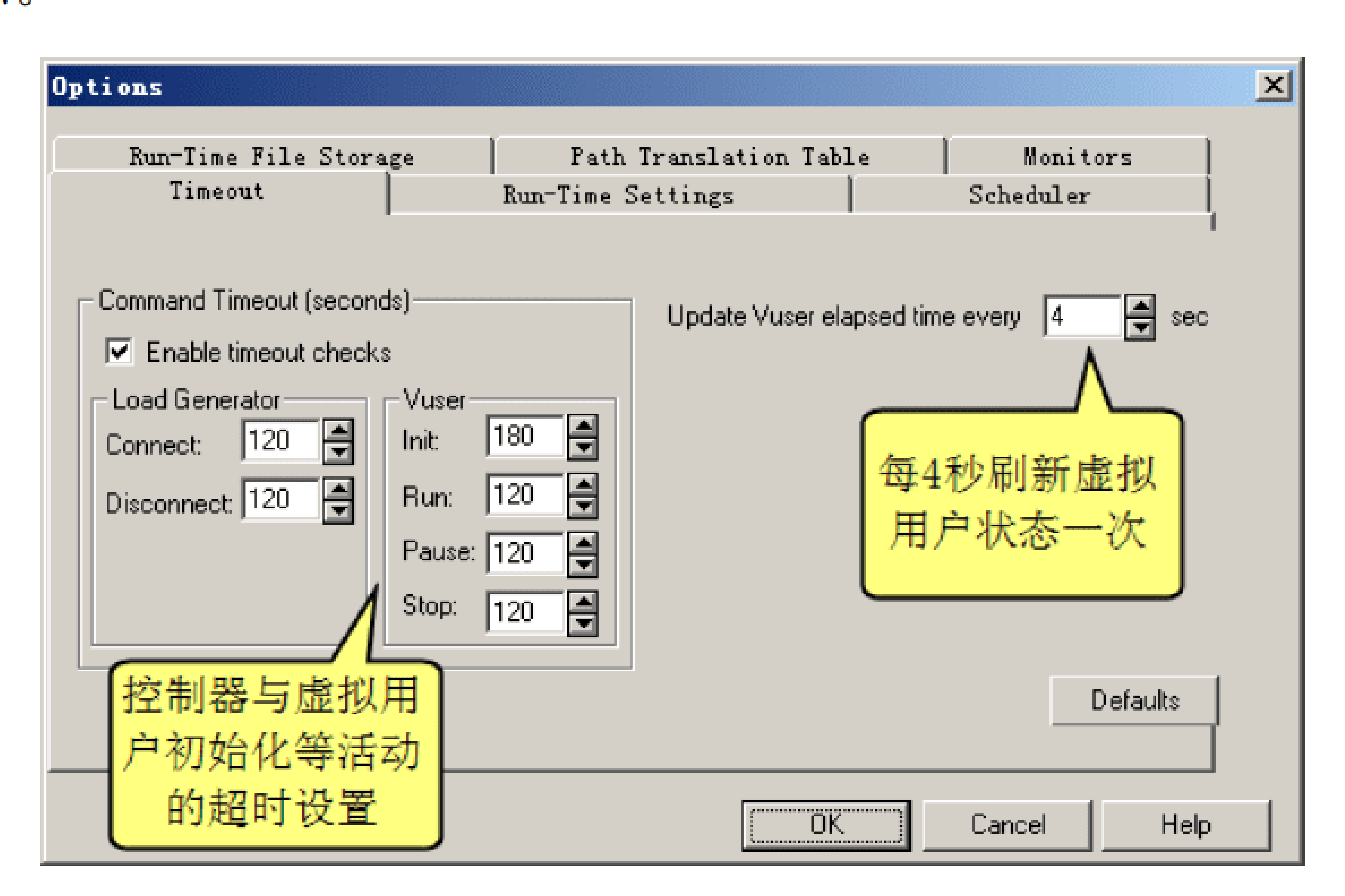


图 10-37　控制器的全局设置窗体

10.4.1　超时设置（Timeout）

图 10-37 是控制器全局设置窗体打开后的界面，同时也可以修改超时的设计，具体需要修改 Timeout 选项卡中的诸多属性。当网络状况不好、场景中机器较多时，可以修改窗体中各个微调框的数值（单位均为秒）以免 LoadRunner 返回错误。

10.4.2　运行时设置（Run-Time Settings）

运行时设置（Run-Time Settings）规定了虚拟用户初始化时的最大数目以及其被停止后场景的行为，如图 10-38 所示。一般情况下，保持系统默认即可。

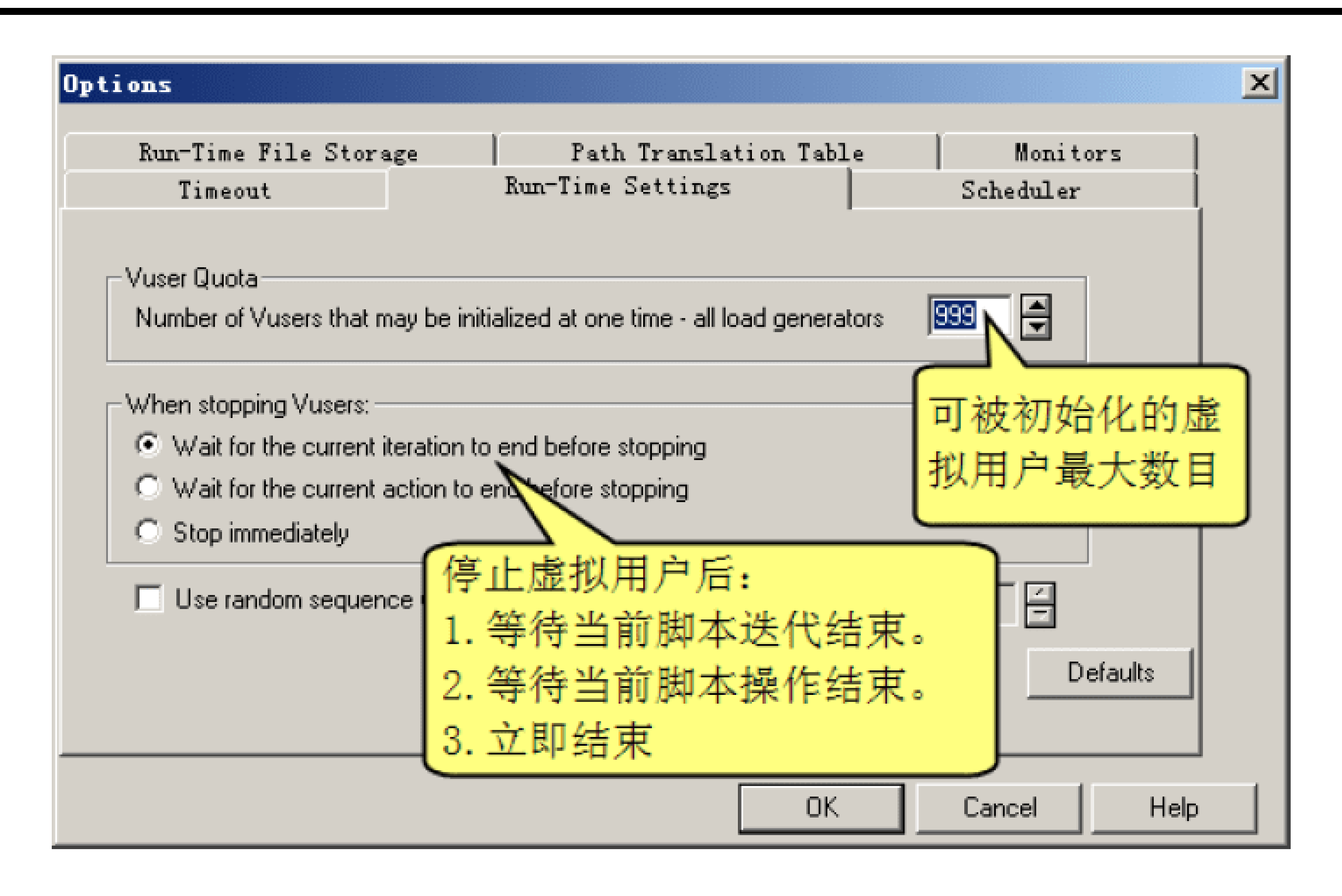


图 10-38　对虚拟用户进行运行时设置

10.4.3　运行时文件存储位置（Run-Time File Storage）

运行时文件存储信息在实际情况中可能需要更改，如图 10-39 所示。

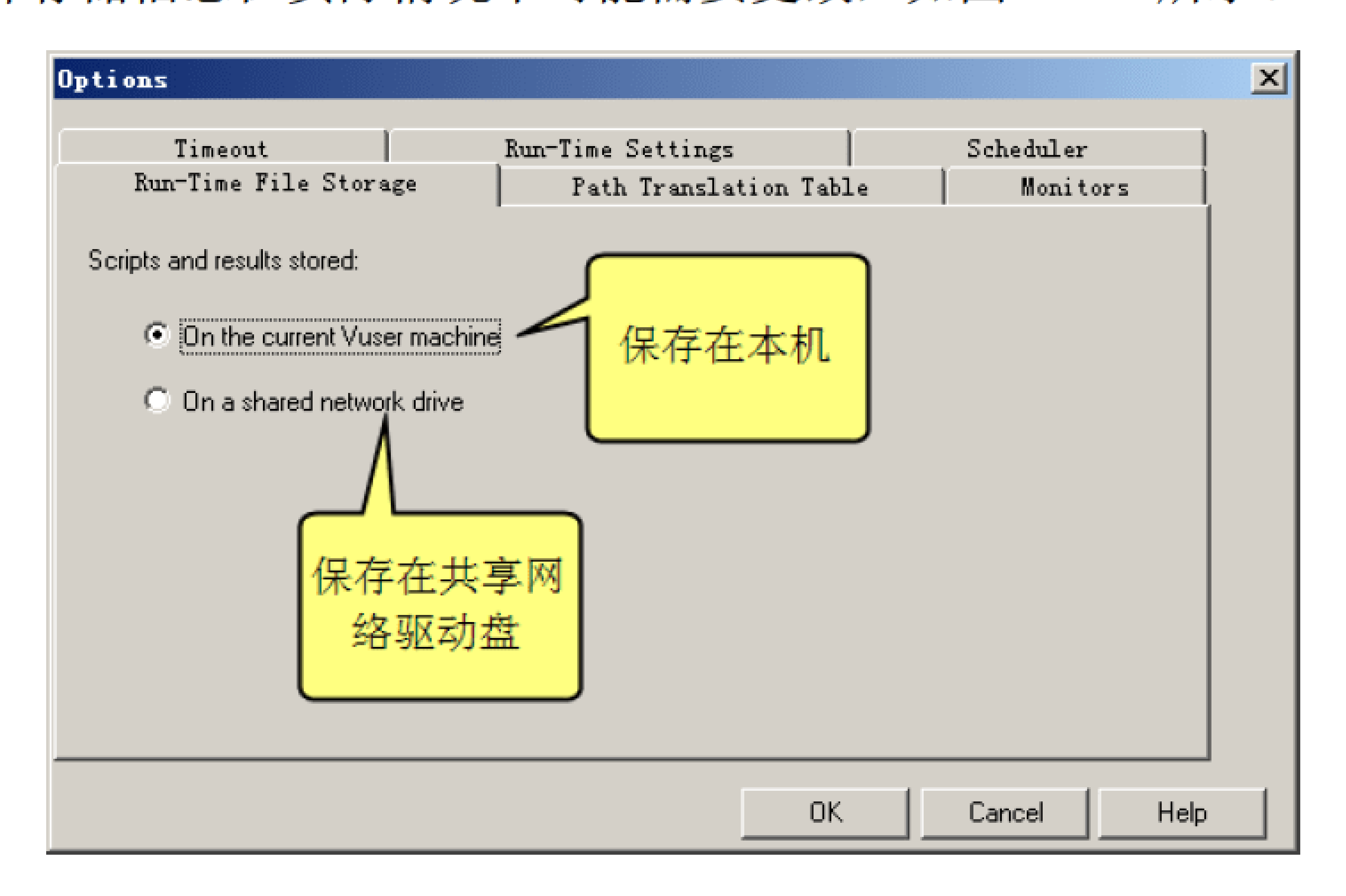


图 10-39　修改运行时文件存储位置

【文件保存位置的选择】

在性能测试的开始阶段，测试工程师需要保证脚本的正确性和稳定性，场景中各组成部分的可靠性，在这个阶段，可以将脚本和运行结果都保存在本机，也就是默认设置。当要进行真正的性能测试之前，最好还是修改此处设置，将脚本和运行结果保存在网络共享当中，利于别人查看和分析。

10.4.4　路径翻译表（Path translation table）

路径翻译表是一种映射，将控制器上的文件路径转换为远程主机上的文件路径。这样的设置对于网络中包含异类系统（比如既有 Windows，也有 Unix）是很有用处的。

LoadRunner 维护一个路径翻译表文件，测试工程师可以在图 10-40 中按照说明修改内容，将控制器的某路径映射为另一台机器的某个文件夹。

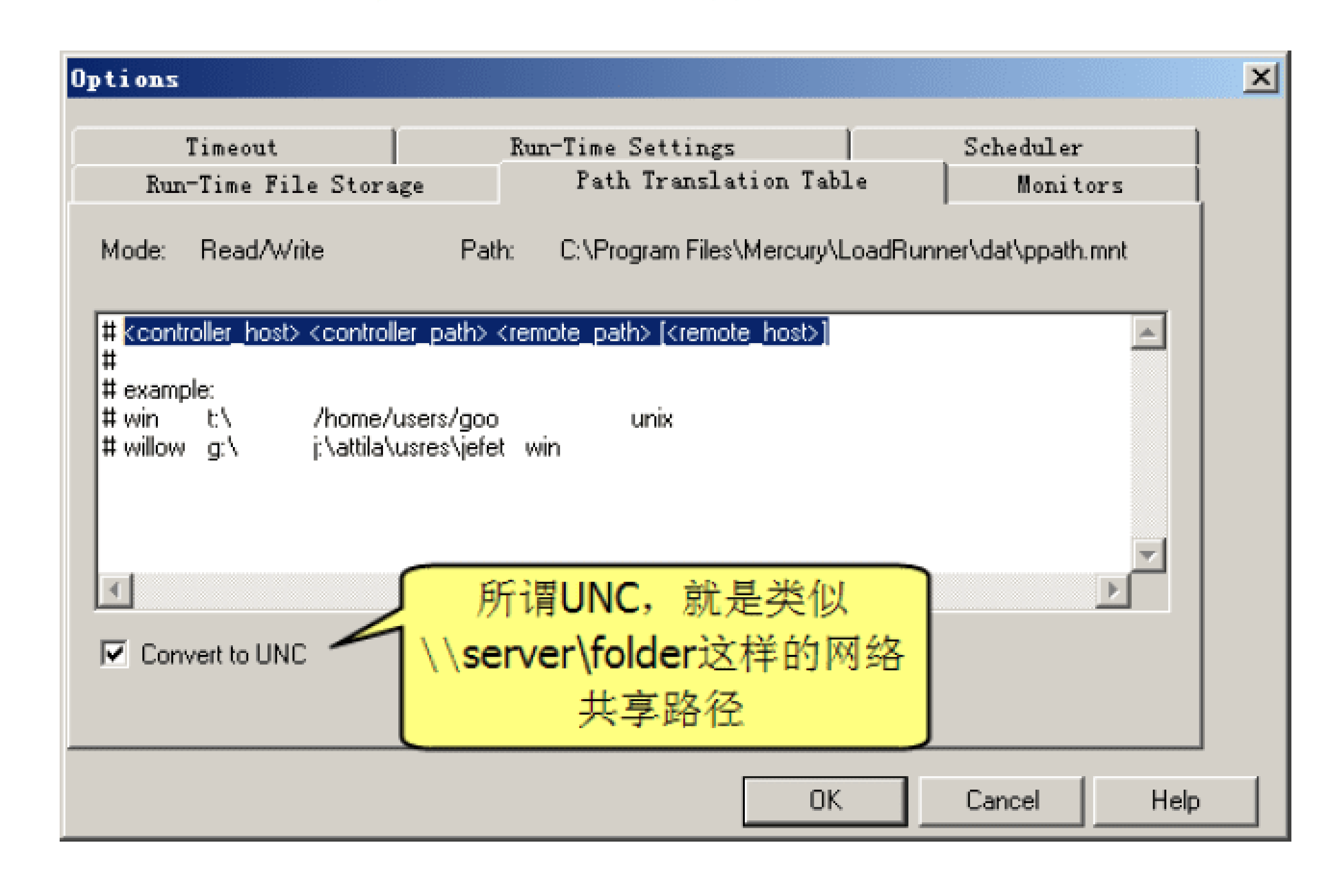


图 10-40　路径翻译表设置

路径翻译表中的每一条记录应该类似如下的语句：

```
<controller_host> <controller_path> <remote_path> [<remote_host>]
```

10.4.5　监视器（Monitors）

监视器好像公路上的电子眼，用于随时报告场景的各项信息，它可以按照规定的时间（也叫做数据采样率，单位为秒）获取服务器资源信息，并反映到控制器界面当中。具体的设置如图 10-41 所示。对于图 10-41 中的出错处理，如果需要性能测试自动运行（比如晚上或者周末工程师不在现场上班的情况），还是选择出错显示在输出窗口为好；否则弹出的对话框可能会中断执行过程。

以上我们简单介绍了控制器的全局设置，实际上，图 10-41 中还有一些全局设置没有显示出来，可以通过选择 Tools|Expert Mode（工具|专家模式）命令来打开，如图 10-42 所示。

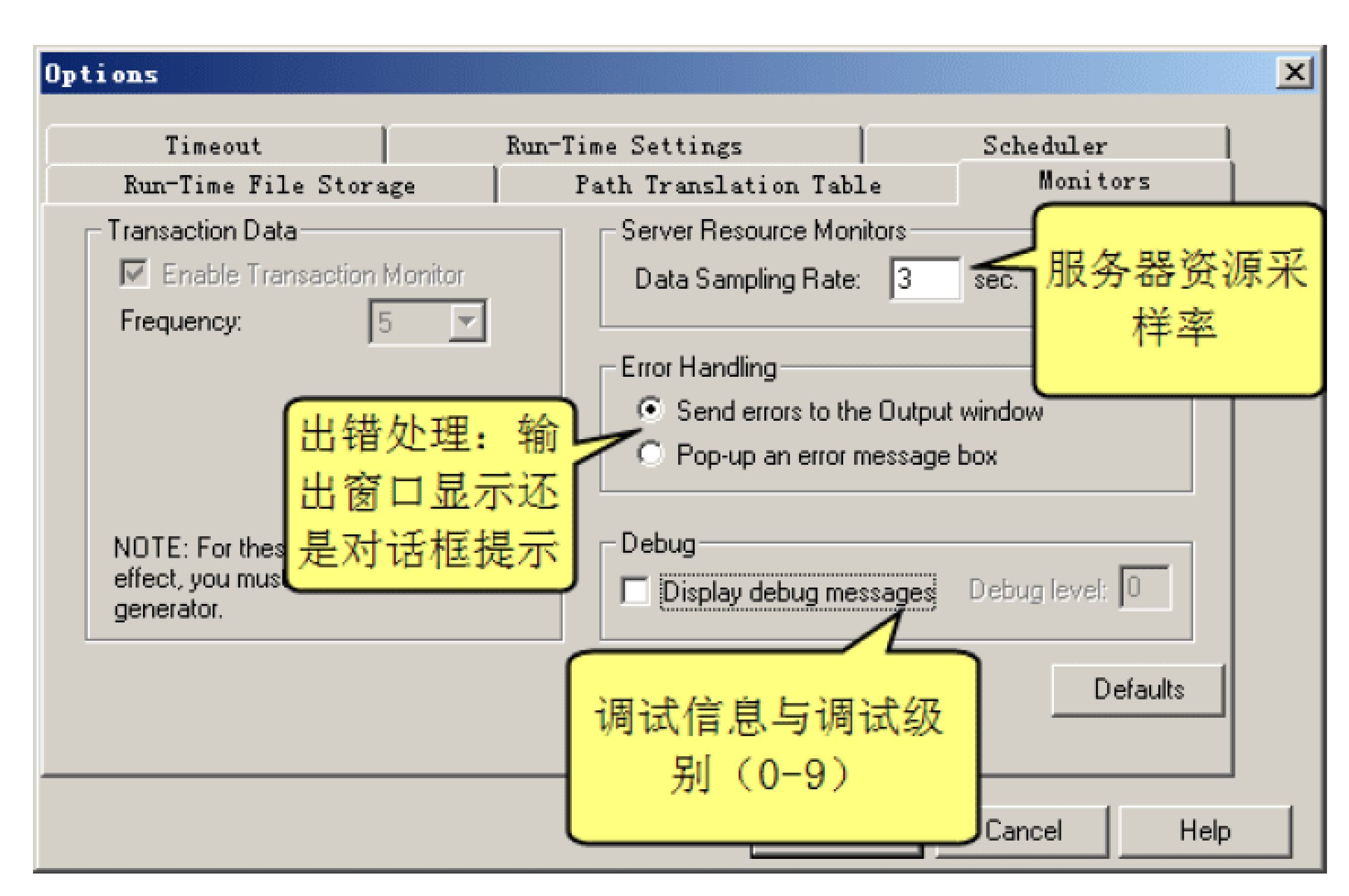


图 10-41　场景的监视器设置

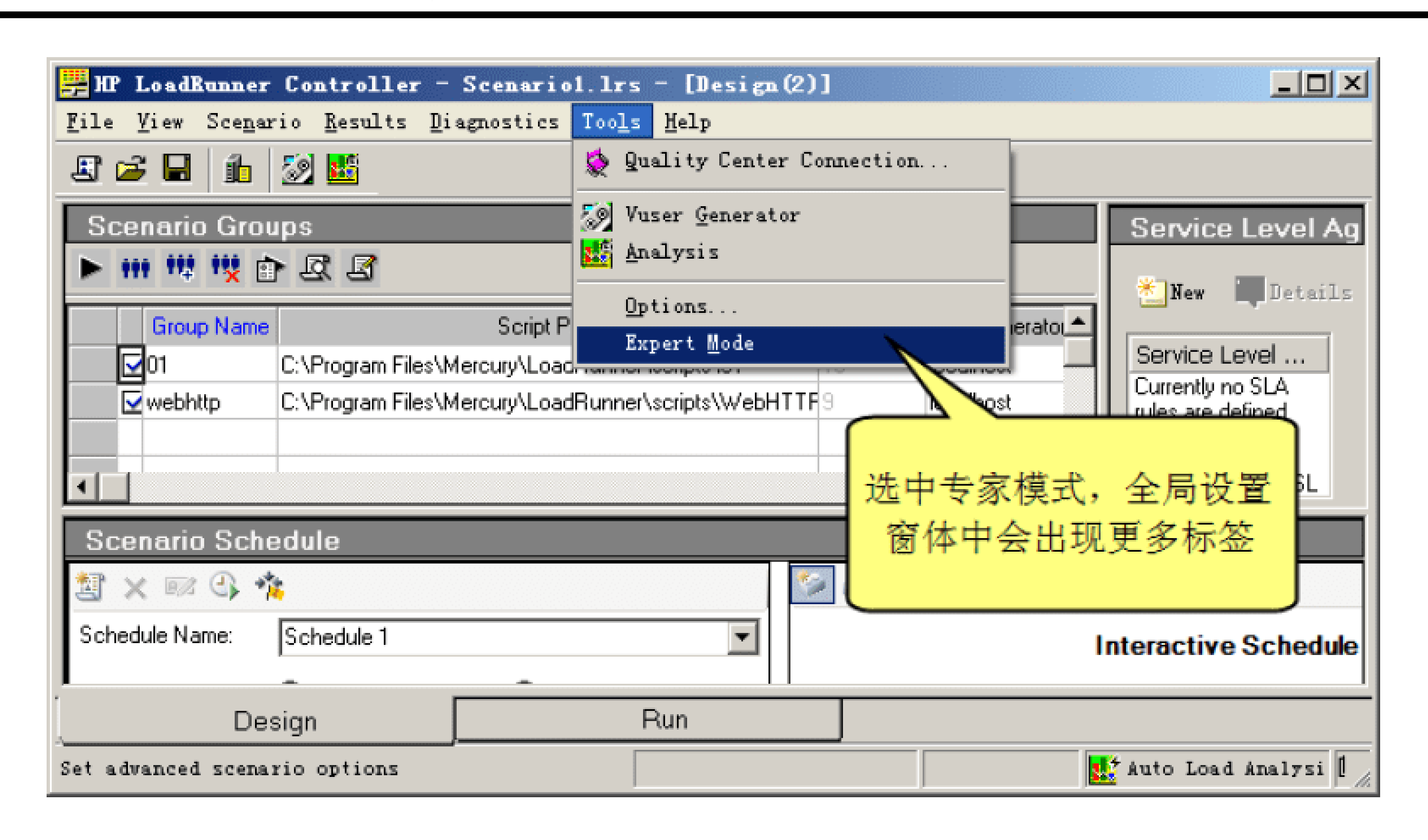


图 10-42　专家模式可以打开更多的全局设置选项

截止这里，有关通过控制器对 LoadRunner 场景进行设置的部分就完成了，实际上还有一些细节需要读者在工作中摸索。有了脚本与用户组组成的场景，测试工程师就可以真正去执行它，从而获得需要的数据。我们将在第 11 章讲解场景的执行与结果分析。

10.5　本 章 小 结

本章介绍了通过 LoadRunner 控制器创建两种类型场景的方法，这两种场景分别是：

- ❑ 面向目标的场景。
- ❑ 手动场景，这也是实际工作中最常用的方法。

场景主要是通过用户的分类以及场景运行的设置来实现的。手动场景中用户的分类需要测试工程师自行设置，主要有用户分组和百分比两种方法。用户经过分组，可以模拟真实的环境，将不同的操作系统、不同的浏览器、不同的特点都考虑到脚本的运行之中。

在场景的设置中一个非常重要的部分就是集合点。所谓集合点就是放置于脚本需要测试并发操作之前的标记，在此时多个用户同时启动开始对被测试软件或者 Web 应用进行操作。场景的运行需要设置执行时间、执行方式（场景还是用户组方式）与运行模式（按真实情况还是按脚本设置执行完毕）。控制器中的场景运行图可以很直观、形象地表达当前场景的运行过程，我们也可以通过修改图中的曲线来达到修改场景操作属性的目的。

场景与场景执行计划建立之后还需要被执行才能获得性能测试的结果，这将是第 11 章的内容。

第 11 章　运行前准备：监控图表与函数

在第 10 章中，我们利用已有的 LoadRunner 脚本创建了一个场景。但是，尚未配置性能测试获取哪些性能计数器：要知道，性能是通过一系列的数据来进行评估的。本章将介绍利用 LoadRunner 控制器（Controller）来进行监控的方法，而监控的对象，正是需要的各种性能计数器。

在配置好监控之后，场景就可以真正意义上的运行了。一般来说，在运行之前，测试工程师有必要对脚本、场景再做一次检查，另外，由于被测试的项目在不同阶段有不同的要求，对录制好的脚本进行修改和完善是很有必要的。通过调用 LoadRunner 提供的一些函数，会非常有利于更好地改进我们的脚本，这也是 LoadRunner 提供的强大功能之一，是初级性能测试工程师提高自身的必经之路。在本章的后半部分将主要介绍函数的使用。

11.1　监控图表与配置

所谓监控，就是 LoadRunner 采集压力生成器上的各种指定性能计数器的过程。监控室发生在性能测试运行过程中的，但是，需要提前设置妥当。在 LoadRunner 中，监控是通过控制器（Controller）界面下方的 Run（运行）选项卡来设置的。

11.1.1　监控与图表

监控和图表是密不可分的。在前文本书介绍过各种操作系统的性能监视器，LoadRunner 的监控也是类似的显示方式。打开监视的方法如图 11-1 所示。

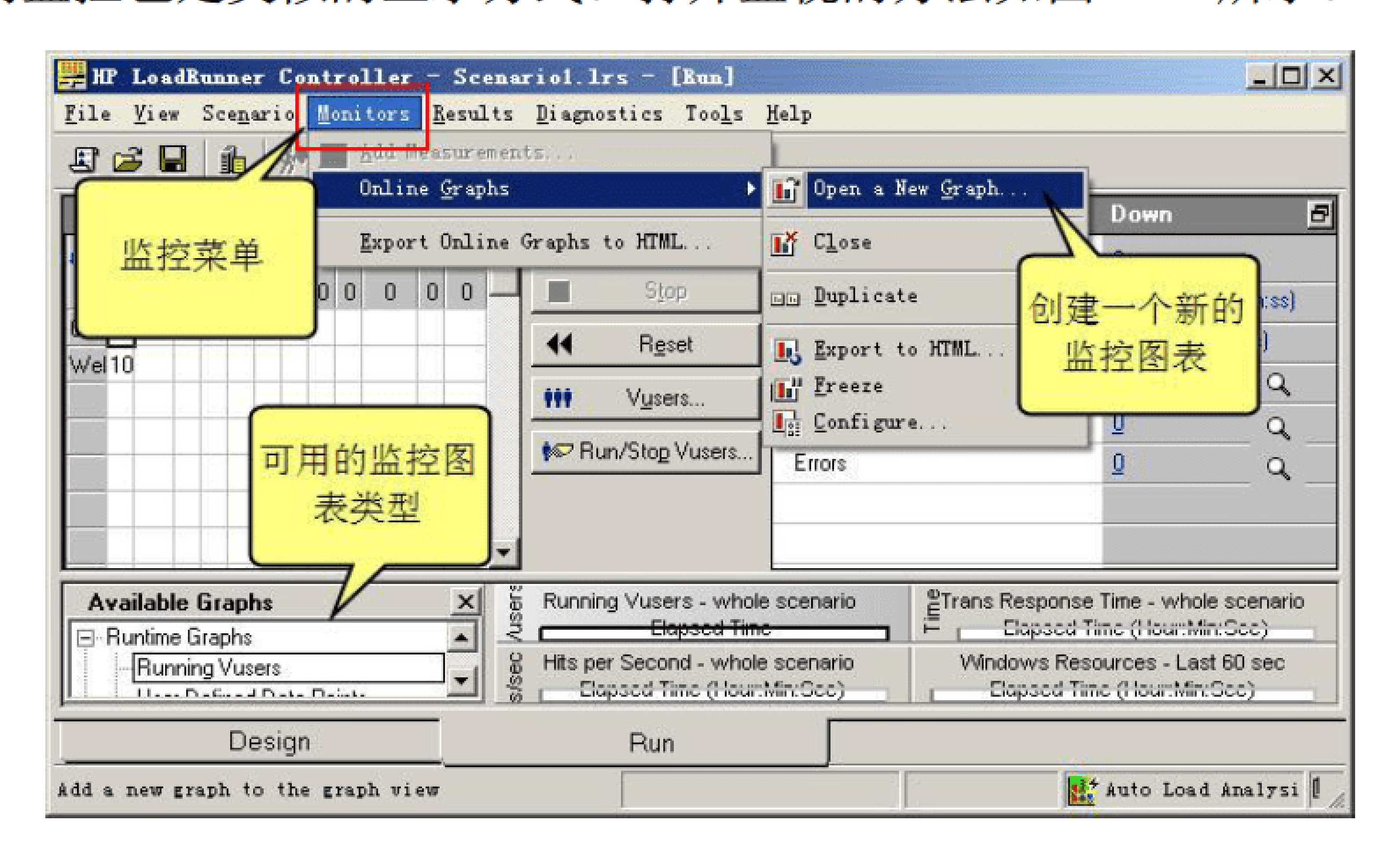


图 11-1　打开/创建一个新的监控图表

在图 11-1 中选择 Open a New Graph（打开新的图表）命令，Controller（控制器）界面将显示目前可用的监视图表列表，如图 11-2 所示。实际上，这些监视类型已经在图 11-1 中左下部分可用图表中显示了出来。

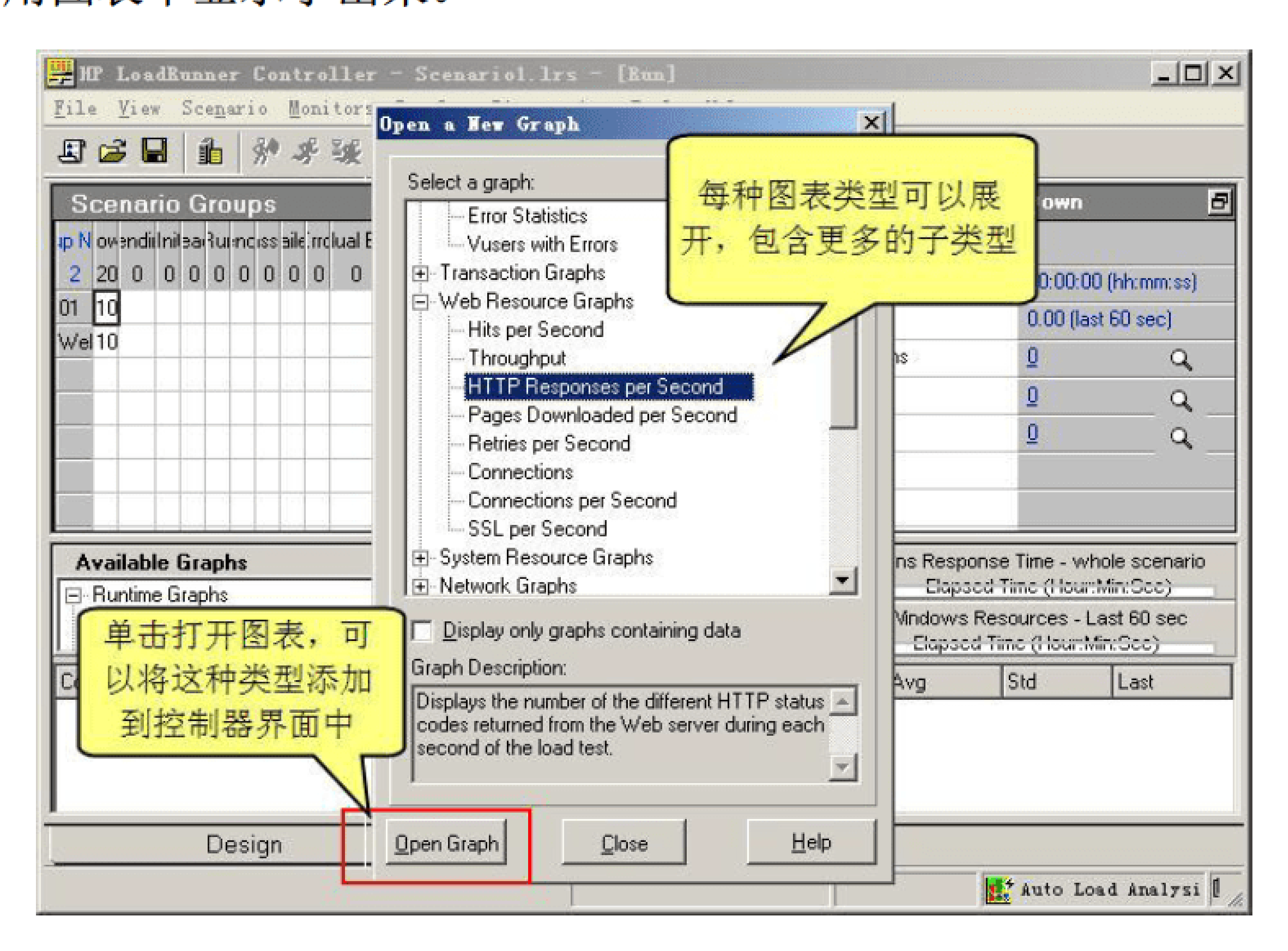


图 11-2　选择待监控的图表类型

【创建图表的含义】

所谓创建图表或者打开图表，实际就是将 LoadRunner 提供的可供监控性能指标类型添加到当前场景的运行当中。此时的控制器处于 Run（运行）选项卡之中，不同于第 10 章场景设置所处的 Design（设计）选项卡，它更类似于地铁车站的调度中心，由工作人员来随时查看目前的运行状况。这也正是监控这个菜单在 LoadRunner 性能测试过程中的含义。

在对 Web 应用进行性能测试时，常用的监视图表类型如下。

- 运行时图表：可以显示运行的虚拟用户、运行时发生的错误、用户自定义数据点信息（User defined data points）等。用户自定义数据可以用 LoadRunner 函数 lr_user_data_point()进行设置，这部分内容将在本章后半部分讲解。
- 交易情况图表：可以显示交易响应时间以及每秒成功交易的数量、百分比等。
- 网络资源情况图表：可以显示每秒点击量、吞吐量、每秒连接数量等。
- 系统资源情况图表：可以显示 Windows、Unix 等不同系统的资源状况等。
- 网络情况图表：显示网络延迟信息。
- 防火墙情况图表：目前只支持某品牌防火墙的信息，在一般场合基本不会用到。
- 网站服务器情况图表：显示多种网站信息服务器（包括 IIS、Apache、iPlanet）的性能信息。
- 网站应用服务器情况图表：显示多种应用服务器（包括 ASP、Websphere、Weblogic 等）的性能信息。
- 数据库情况图表：显示 4 种主流数据库（DB2、Oracle、SQLServer、Sybase）的性能信息。
- 流媒体服务器情况图表：可以显示 Windows Media Server 和 Real Server 以及相应

客户端的性能信息。

- 其他类型图表：中间件、CRM/ERP 等，由于涉及更多计算机应用领域，使用方法也没有大的不同，本书就不做介绍了。

11.1.2　对运行状况、交易状况进行监控

在图 11-2 左侧 Available Graphs（可用图表）视图中展开 Runtime Graphs（运行图）节点，即可选择其中某一种类型添加至控制器运行标签的界面，如图 11-3 所示。读者可以看到，在图中，总共添加了多个不同类型的图表进行监控。在图表显示的空白区域右击，在弹出的快捷菜单中选择 Configure（配置）选项，即可弹出窗体，对所有图表的属性进行设置，如图 11-4 所示。需要特别说明的是如下几个设置：

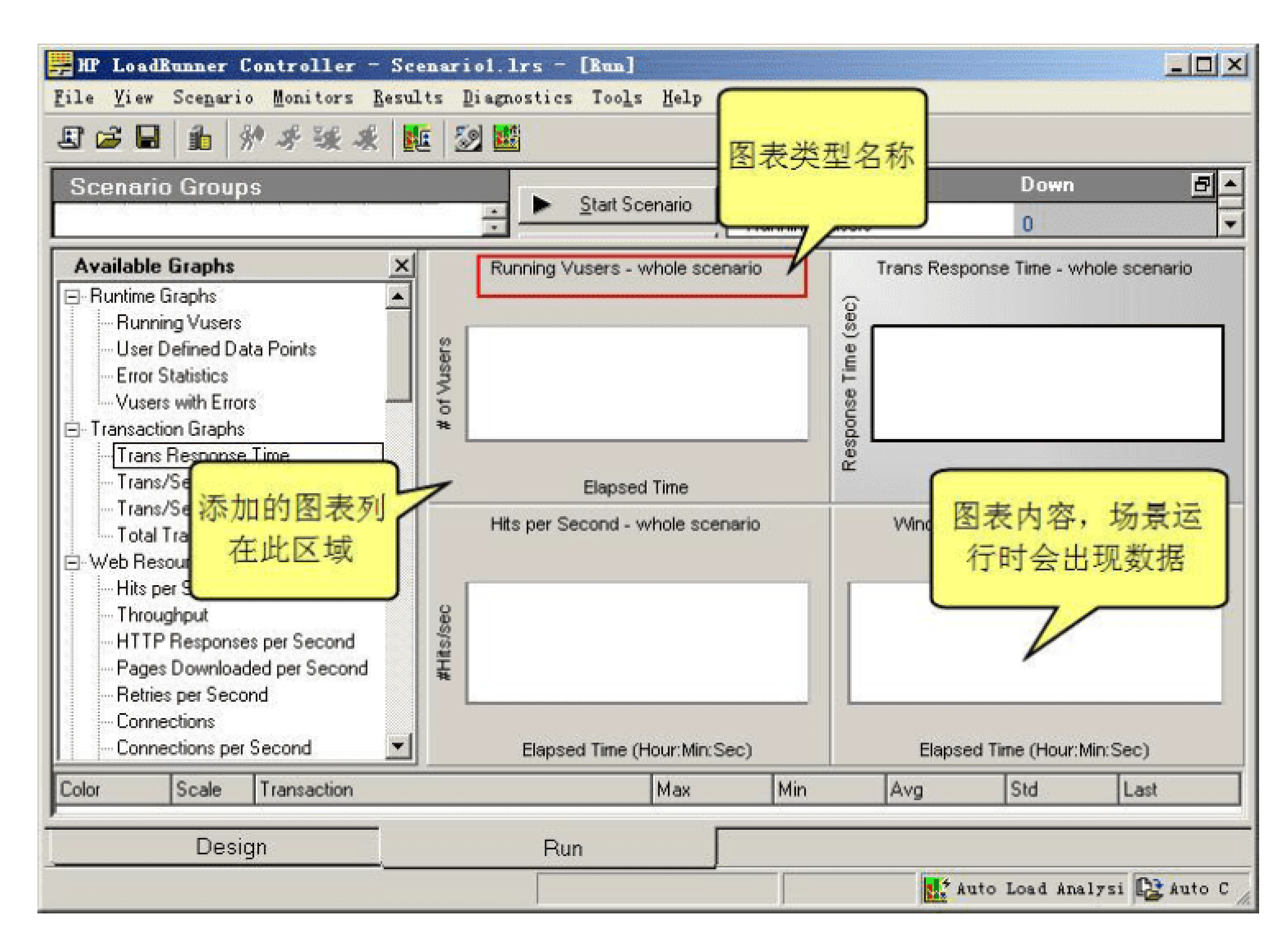


图 11-3　添加图表后控制器的界面

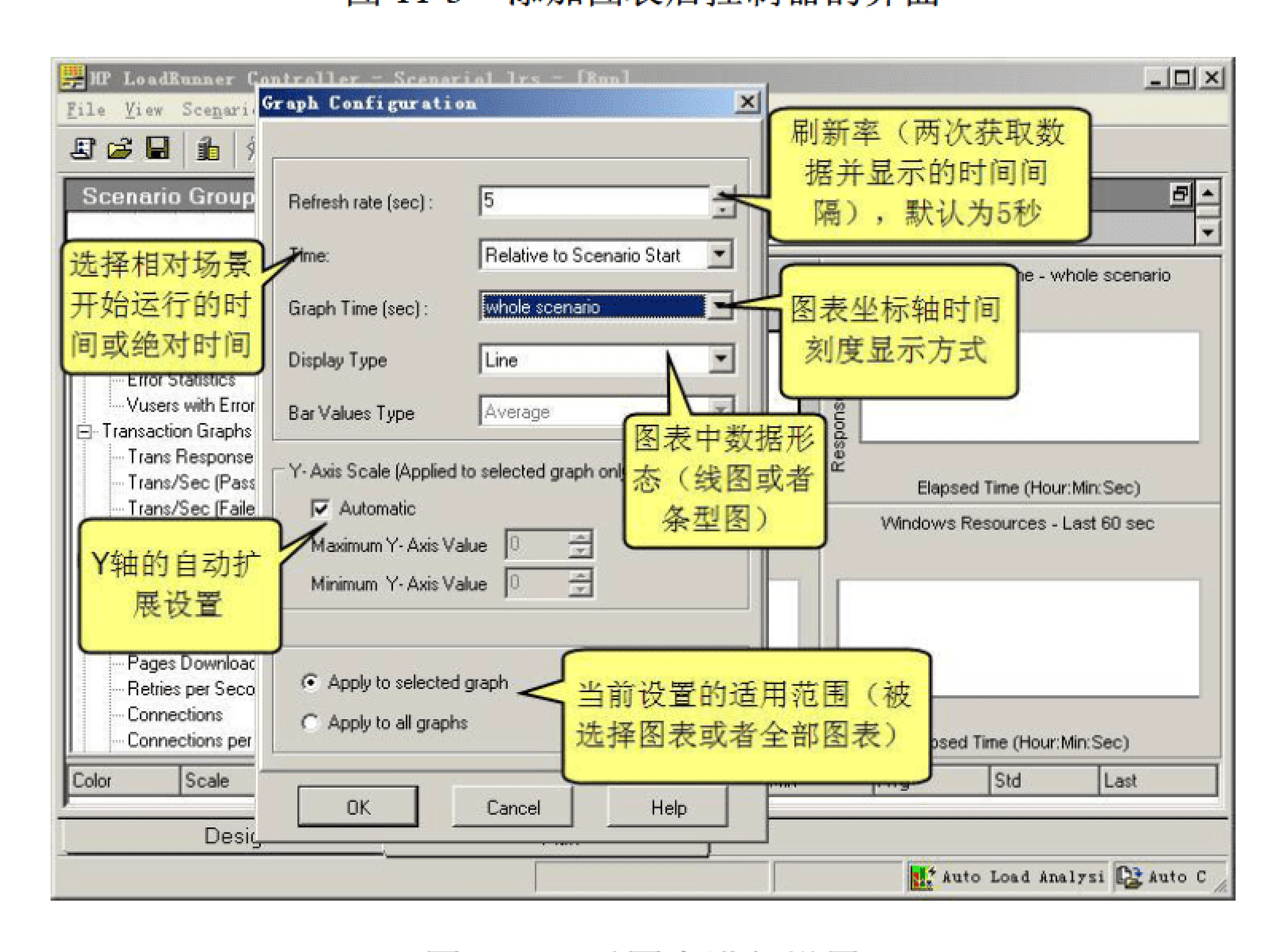


图 11-4　对图表进行设置

（1）Graph Time（图表时间）下拉列表框：该菜单能设置图表中的时间作为 X 轴（一般时间均为 X 轴）的显示方式。当设置为 Whole scenario 时，图表显示整个场景中被关注性能计数器的变化情况。当设置为其他具体数值（有 60、80、3600 等多项选择）时，图表在运行时间超过该数值后，将擦除图表之前的数据重新显示，这种方式类似于我们在资源管理器中看到的实时资源显示。

（2）Y 轴的自动扩展设置（Y-Axis Scale）：在场景的实际运行中，每一时刻获取的数据大小不是完全可以预知的，因此，数据之间的差别会比较大，为了显示图表的可读性，Y 轴的坐标有必要根据数值进行调整，这可以通过 Automatic（自动化）复选框来实现。

（3）设置完毕后单击 OK 按钮，更改后的图表属性生效。

【增加度量值】

另外，我们还可以在某一个性能图表的空白区域右击，在弹出的快捷菜单中选择 Add Measurements（增加度量）选项来设置该图表监控哪些计算机的性能，如图 11-5 所示。该对话框也可以通过依次选择控制器（Controller）的 Monitors|AddMeasurements（监控器|增加度量）命令打开。不过，值得注意的是，并不是所有的图表都有这个选项，当选项不可用时为灰色。

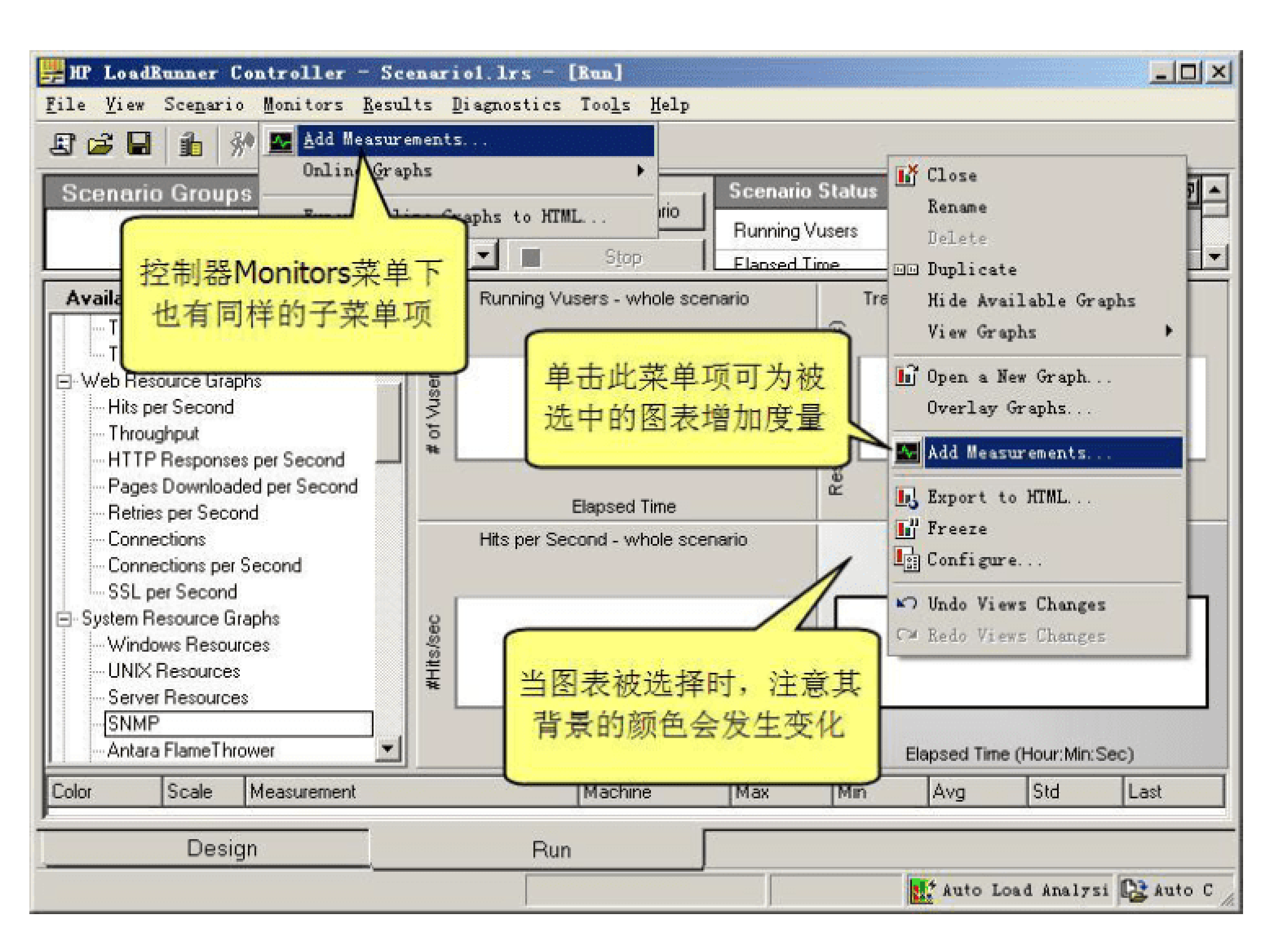


图 11-5　为图表增加度量的菜单入口

当单击增加度量之后，控制器将弹出一个对话框，如图 11-6 所示。在对话框中可以通过单击 Add 按钮加入新的被监控机器；同时，对话框下半部分还将列出现有被监视机器上各性能计数器度量的清单。

对于本章后面将要介绍的各种性能图表，都可以施行如上的属性设置操作，方法相同，在下文就不再赘述了。

11.1.3　对系统与网络资源进行监控

在图 11-2 所示的列表中选择系统图表，即可以对整个操作系统进行监控。该项目下有几个选择，分别支持获取 Windows、Unix、服务器、SNMP、Antara FlameThrower 和 SiteScope

（后两者我们日常碰到的机会较少，分别是 Antara 公司生产的专用网络设备与惠普的专业监控软件）的一些性能计数器数值并显示在图表上。

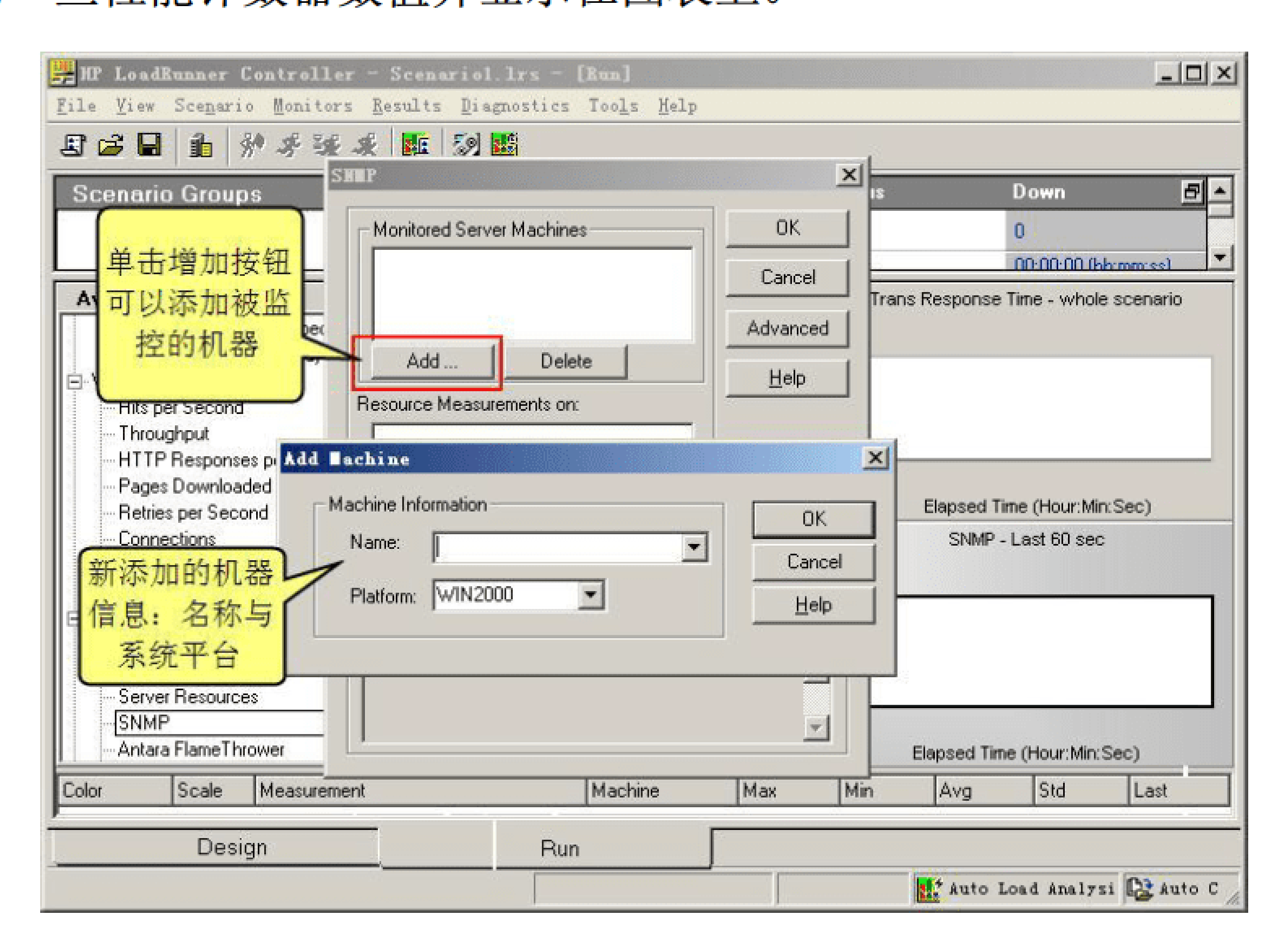


图 11-6　为图表添加被监控机器

对于 Windows 系统，可以监视如表 11-1 所列出的性能计数器。具体各计数器的详情也可以参看本书前面章节的介绍。

表 11-1　Windows系统监控图表所获取的性能计数器信息

性 能 对象	计　数　器	数 值 说 明
System	% Total Processor Time	表明系统中所有处理器工作时间百分比。如果有多个 CPU，该数值是总平均数。例如：系统内一共 2 块 CPU，1 块 100%处于忙碌，另 1 块 10%忙碌状态，则该数值为 55%
Processor	% Processor Time	（Windows 2000）表明当前处理器处理特定线程所花费的时间百分比
System	File Data Operations/sec	整个系统每秒钟对文件系统进行纯数据读写的操作次数，注意这里并不包括控制读写数据操作所花费的额外操作
System	Processor Queue Length	等待处理器处理的队列长度，数值为线程的数量，如果超过 2，则说明 CPU 出现瓶颈。这个数值不包括正在处理的线程数，只是等待处理的线程数量
Memory	Page Faults/sec	每秒内存中发生的页面错误
PhysicalDisk	% Disk Time	物理磁盘处理读写操作占规定时间的百分比
Memory	Pool Nonpaged Bytes	非页面池的大小（字节数）
Memory	Pages/sec	每秒页面数量
System	Total Interrupts/sec	系统每秒总中断数量
Objects	Threads	线程数量
Process	Private Bytes	特定进程独享的数据区域字节数

而对于 Unix 系统图表，则可以监视如表 11-2 所列出的数据。

表 11-2　Unix系统监控图表所获取的性能计数器

计　数　器	数 值 说 明
Average load	系统刚过去的 1 分钟内就绪的进程数量
Collision rate	以太网每秒的碰撞率
Context switches rate	每秒钟线程或进程进行上下文切换的次数
CPU utilization	CPU 利用时间百分比
Disk rate	磁盘读写操作的百分比
Incoming packets error rate	以太网接收数据包的错误率
Incoming packets rate	每秒钟以太网接收的数据包
Interrupt rate	每秒钟设备中断数量
Outgoing packets errors rate	以太网发送数据包的错误率
Outgoing packets rate	每秒钟以太网发送的数据包
Page-in rate	每秒钟装入内存的页面数量
Page-out rate	每秒钟内存读出并保存至页面文件的数量
Paging rate	每秒钟页面处理（包括出和入）的数量
Swap-in rate	进程交换时写入缓存的数量
Swap-out rate	进程交换时读出缓存的数量
System mode CPU utilization	系统模式下 CPU 利用率
User mode CPU utilization	用户模式下 CPU 利用率

对于服务器监控图表，则是对 Unix 监控的补充，提供了如下 3 种关于磁盘的信息，如表 11-3 所示。

表 11-3　服务器监控图表所能获取的性能计数器

监　　控	性能计数器	数 值 说 明
CPU Monitor	Utilization	CPU 利用率
Disk Space Monitor	Disk space	磁盘可用与剩余空间情况
Memory Monitor	MB free	可用内存数量，以 MB 为单位
	Pages/sec	每秒页面文件在内存与文件系统之间交换数量
	Percent used	内存使用百分比
Services Monitor		监控本地或远程机器中某服务是否成功运行

【有关 SiteScope】

SiteScope 是惠普公司开发的用于监视由多台服务器构成的系统性能的软件。在 LoadRunner 中，监控功能是通过 Monitor Engine（监控引擎）来实现数据获取的。除了系统默认的监控引擎之外，如果在读者的工作环境中安装有 SiteScope，也可以将 LoadRunner 的默认监控引擎设置更改为使用 SiteScope。具体操作方法则是在图 11-6 中单击 Advanced（高级）按钮，弹出对话框如图 11-7 所示，根据实际情况修改即可。

11.1.4　对防火墙、网络服务器进行监控

目前 LoadRunner 对于防火墙监控只支持 Checkpoint 公司的 Firewall-1 产品，在实际工

作中应用并不广泛，也需要网络安全和网络管理工程师的协同参与，本书就不做详细介绍了，读者可以查阅相关的书籍和使用手册。

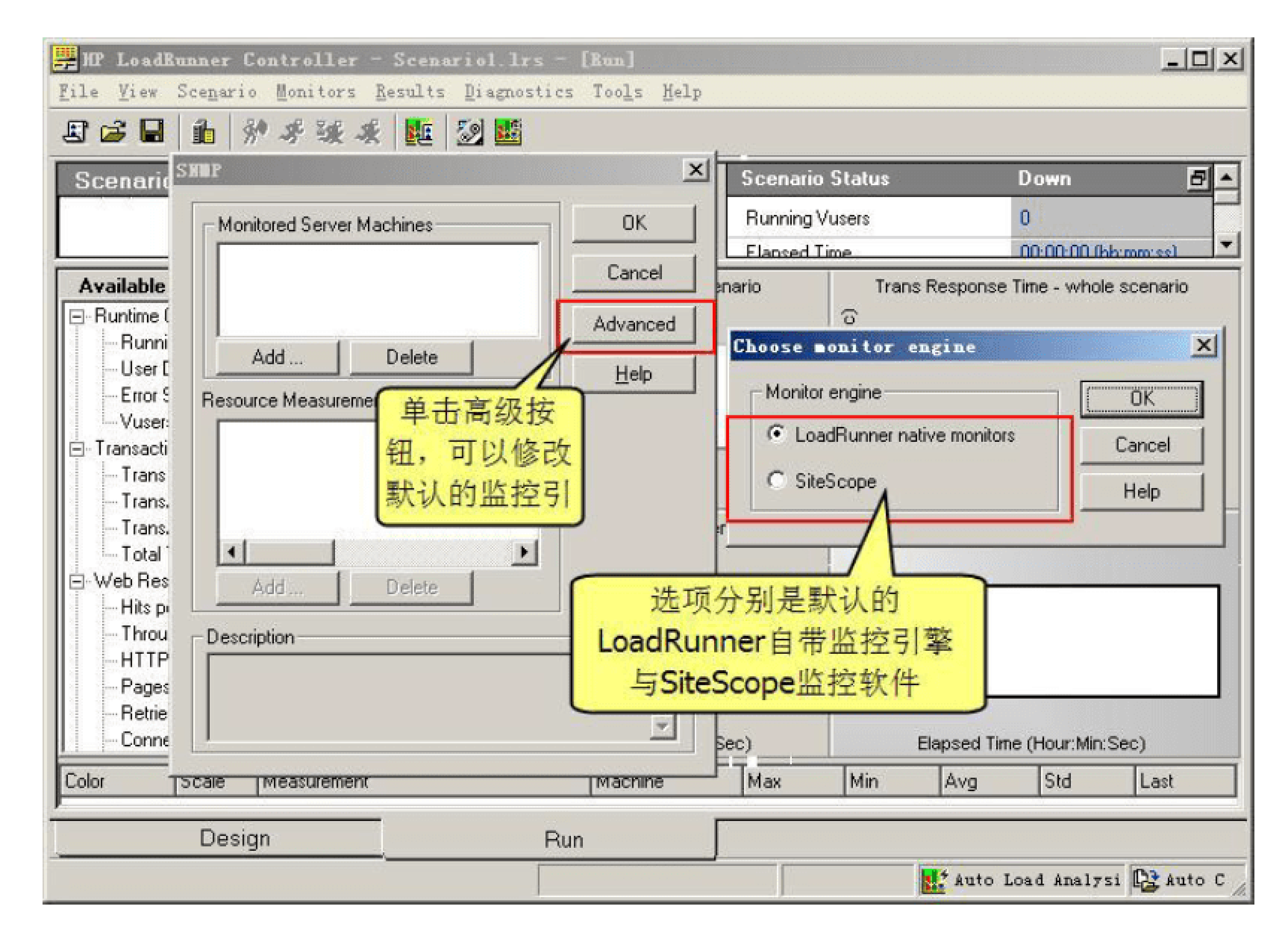


图 11-7　针对某性能图表修改 LoadRunner 默认的监控引擎

对于网络服务器，LoadRunner 对于微软的 IIS，可以监控到表 11-4 列出的性能计数器。

表 11-4　微软IIS的监控图表

对　　象	性能计数器	说　　明
Web Service	Bytes Sent/sec	Web service 每秒发送字节数
Web Service	Bytes Received/sec	Web service 每秒接收字节数
Web Service	Get Requests/sec	Web service 每秒 Get 请求数量
Web Service	Post Requests/sec	Web service 每秒 Post 请求数量
Web Service	Maximum Connections	Web service 截至目前最大同时连接数量
Web Service	Current Connections	Web service 当前连接数量
Web Service	Current NonAnonymous Users	Web service 当前非匿名用户数量
Web Service	Not Found Errors/sec	每秒请求文档未找到的数量（HTTP 404 错误）
Process	Private Bytes	网络服务器享有的私有数据块字节大小

对于另一种主流的网络服务器 Apache，LoadRunner 监控图表可以获取表 11-5 列出的性能计数器。

表 11-5　Apache监控图表可获取的性能计数器

性能计数器	说　　明
# Busy Servers	处于忙碌（Busy）状态的服务器数量
# Idle Servers	处于空闲（Idle）状态的服务器数量
Apache CPU Usage	Apache 所占用 CPU 情况
Hits/sec	每秒 HTTP 请求。（至少每秒有一次请求，该请求来自监控引擎）
KBytes Sent/sec	网络服务器所发送的 KB 字节数

11.1.5　对中间件进行监控

对于中间件，LoadRunner 目前支持 Tuxedo 与 IBM 的 WebSphere MQ。与之前的若干监控不同，中间件的监控需要在控制器（LoadRunner Controller）所在机器上安装客户端软件，从而能够连接以获取相关性能数据。对于添加 Tuxedo 性能监控，主要过程如下：

- ❑ 在 LoadRunner 控制器所在机器上安装 Tuxedo 工作站的客户端。
- ❑ 确认该机器上 TUXDIR 环境变量是否正确。如果不正确，将弹出如图 11-8 所示的错误信息。一般来说，安装好客户端，该环境变量也会同时设置好。

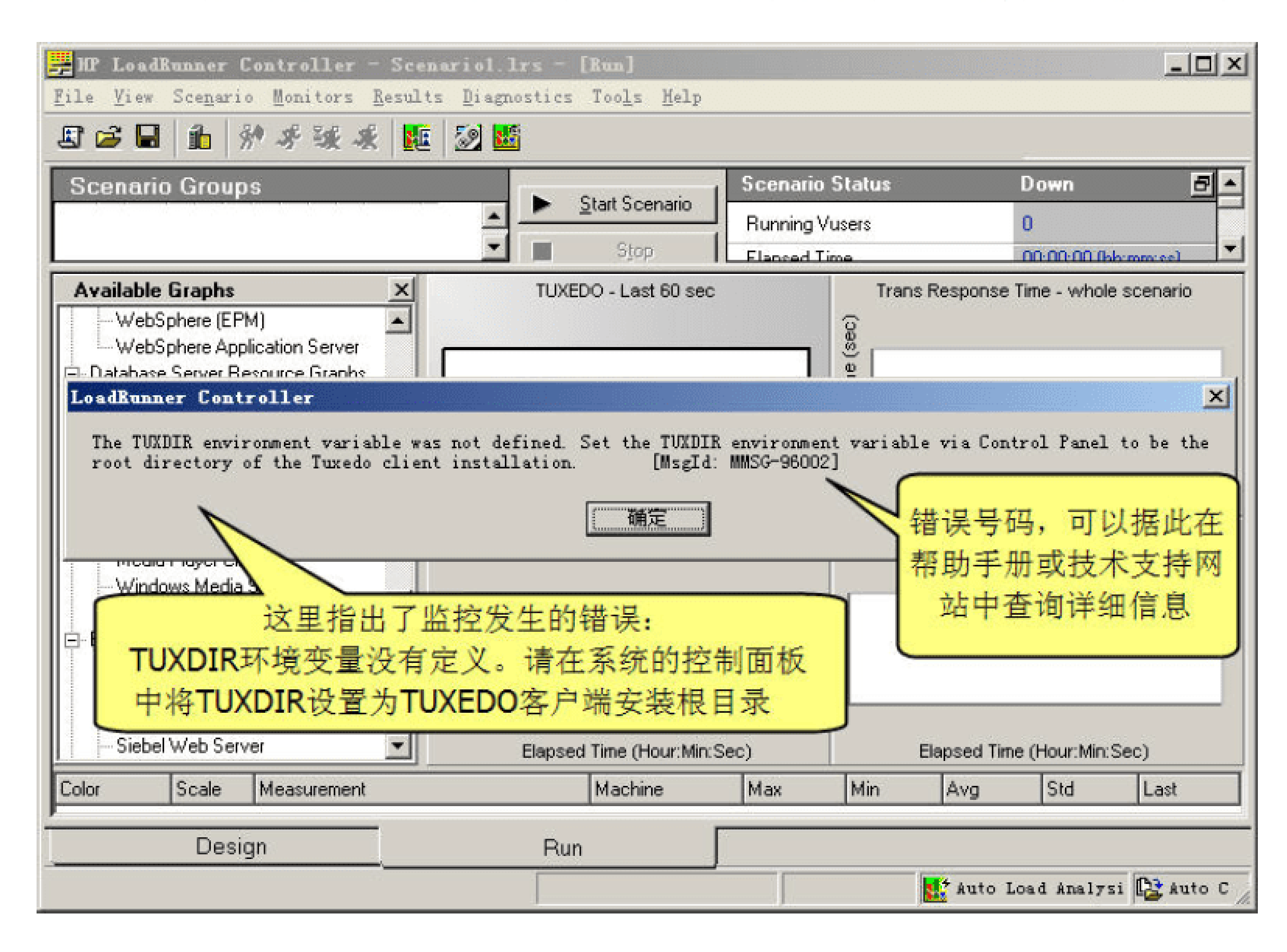


图 11-8　设置 TUXEDO 监控需要安装客户端

- ❑ 配置 Tuxedo 服务器，在服务器端启动工作站监听（WSL）服务，并设置好端口号。客户端将通过此端口与 Tuxedo 服务器进行通信，因此要注意不要与其他应用所使用端口重复。
- ❑ 设置连接 Tuxedo 的用户名、密码、服务器地址与端口以及客户端名称，通过图 11-5 所显示的增加度量对话框添加监控。

Tuxedo 监控图表可以默认提供表 11-6 所列出的性能计数器。

表 11-6　Tuxedo监控图表可提供的默认性能计数器

对　　象	性能计数器	说　　明
Machine	% Busy Clients	当前正在等待服务器响应的已登录活动客户端数量所占百分比
	Active Clients	当前已登录的活跃客户端总数量
	Busy Clients	当前正在等待服务器响应的已登录活动客户端数量
	Current Accessers	当前已直接或通过工作站处理程序访问 Tuxedo 服务器应用程序的客户端与服务器总数量
	Current Transactions	当前计算机中正发生的事务表记录

续表

对　　象	性能计数器	说　　明
Machine	Idle Clients	当前未等待服务器响应的已登录活动客户端数量
	Workload Completed/second	每秒所有服务器所完成的总工作负荷
	Workload Initiated/second	每秒所有服务器所启动的总工作负荷
Queue	% Busy Servers	当前正在处理 TUXEDO 请求的活跃服务器占总量的百分比
	Active Servers	当前正在处理或等待处理 TUXEDO 请求的活跃服务器数量
	Busy Servers	当前正在处理 TUXEDO 请求的活跃服务器数量
	Idle Servers	当前等待处理 TUXEDO 请求的活跃服务器数量
	Number Queued	已经处于队列中的消息总数
Server	Requests/second	每秒处理的服务器请求
	Workload/second	每秒处理的工作负荷。注意：工作负荷是服务器请求的加权度量，默认情况下，工作负荷是请求的 50 倍，但有些请求拥有不同的权重
Workstation Handler (WSH)	Bytes Received/sec	工作站处理程序每秒接收到的字节总数
	Bytes Sent/sec	工作站处理程序每秒发送的字节总数
	Messages Received/sec	工作站处理程序每秒接收到的消息总数
	Messages Sent/sec	工作站处理程序每秒发送的字节总数
	Number of Queue Blocks/sec	每秒工作站处理程序所用队列发生阻塞次数，通过该数值可以对工作站处理程序是否过载有大致判断

对于中间件的详细监控，还需要性能测试工程师与其他相关人员在实际情况下的协同工作，以保证测试过程与结果的客观、合理与准确。

11.1.6　对数据库进行监控

对于绝大部分 Web 应用来说，都需要后端数据库的支持。因此，获取数据库的性能情况非常重要。

LoadRunner 支持主流数据库的性能监控。下面我们以微软的 SQL Server 为例，讲解数据库监控的一般过程。

- 在控制器所在计算机或专门的 Windows 机器上安装配置 SQL Server 客户端程序，这将安装 SQL Server 通信等功能的相关文件。
- 在控制面板 | 服务中确认远程过程调用和远程注册表等 3 个服务处于运行状态。这 3 个服务的英文名称如下。
  - Remote Registry：远程注册表；
  - Remote Procedure Call Locator：远程过程调用定位器；
  - Remote Procedure Call：远程过程调用。
- 添加 SQL Server 监控图表，为其增加度量。

SQL Server 监控图表默认可以提供如表 11-7 列出的性能计数器。

表 11-7　SQL Server监控图表默认获取的性能计数器

性能计数器	说　明
% Total Processor Time (NT)	所有处理器处理非空闲（Idle）进程所花费时间占总时间的平均百分比
Cache Hit Ratio	在缓冲区高速缓存中找到而不需要从磁盘中读取的页的百分比。该比率是缓存命中总次数与过去几千页访问以来的缓存查找总次数之比。经过很长时间后，该比率的变化一般会很小。由于从缓存中读取数据比从磁盘中读取数据的开销小得多，普遍希望该比率高一些会好。如果过低，则可以通过增加分配给 SQL Server 的内存数量来解决
I/O - Batch Writes/sec	每秒批量写入磁盘的缓冲区数量
I/O - Lazy Writes/sec	每秒被缓冲区管理器的惰性编写器写入的缓冲区数。惰性编写器（Lazy Writer）是一个 SQL Server 的系统进程，用于成批刷新脏的老化的缓冲区（包含更改的缓冲区，必须将这些更改写回磁盘，才能将缓冲区重用于其他页），并使得这些数据可用于用户进程。惰性编写器不需要为创建可用缓冲区而频繁执行检查点（SQL Server 的 Checkpoint，与 LoadRunner 中的同名词无关）
I/O - Outstanding Reads	单位时间等待读取数据数量
I/O - Outstanding Writes	单位时间等待写入数据数量
I/O - Page Reads/sec	每秒页面读取数量
I/O - Transactions/sec	每秒事务处理数量
User Connections	当前打开的用户连接数量
% Processor Time (Win 2000)	当前处理器处理非空闲（Idle）进程所花费的时间占总量的百分比

对于其他类型的主流数据库，各自有不同的性能计数器，但是添加监控图表的过程类似，感兴趣的读者可以在工作中进行一番尝试和探索。

11.1.7　监控图表的常见操作技巧

前面我们学习了在 LoadRunner 的控制器中，如何添加与配置与 Web 应用相关的软硬件性能监控图表。在使用监控图表的过程中，也有一些很方便的技巧，在本节中将做简单介绍。

【双击监控图表】

通过在某个监控图表空白处双击，可以将当前的图表放大以充满整个图表显示区域。如果再次双击，则图表会恢复为原来的位置和大小。这一点在分析图表阶段，需要对图表中各数据进行详细查看的时候会非常有用。

【设置多图表显示】

LoadRunner 控制器默认的图表显示区域可以显示 4 个（可以是相同类型、也可以是不同类型）性能监控图表。但是，如果工作中需要显示更多的情况下该如何处理呢？答案也很简单：在任意一个图表空白处右击，在弹出的快捷菜单中选择 View Graphs（查看图表）选项，在弹出的快捷菜单中选择 1、2、4、8 这几个数量或者 Custom Number（自定义数量）设置其他数值即可，如图 11-9 所示。

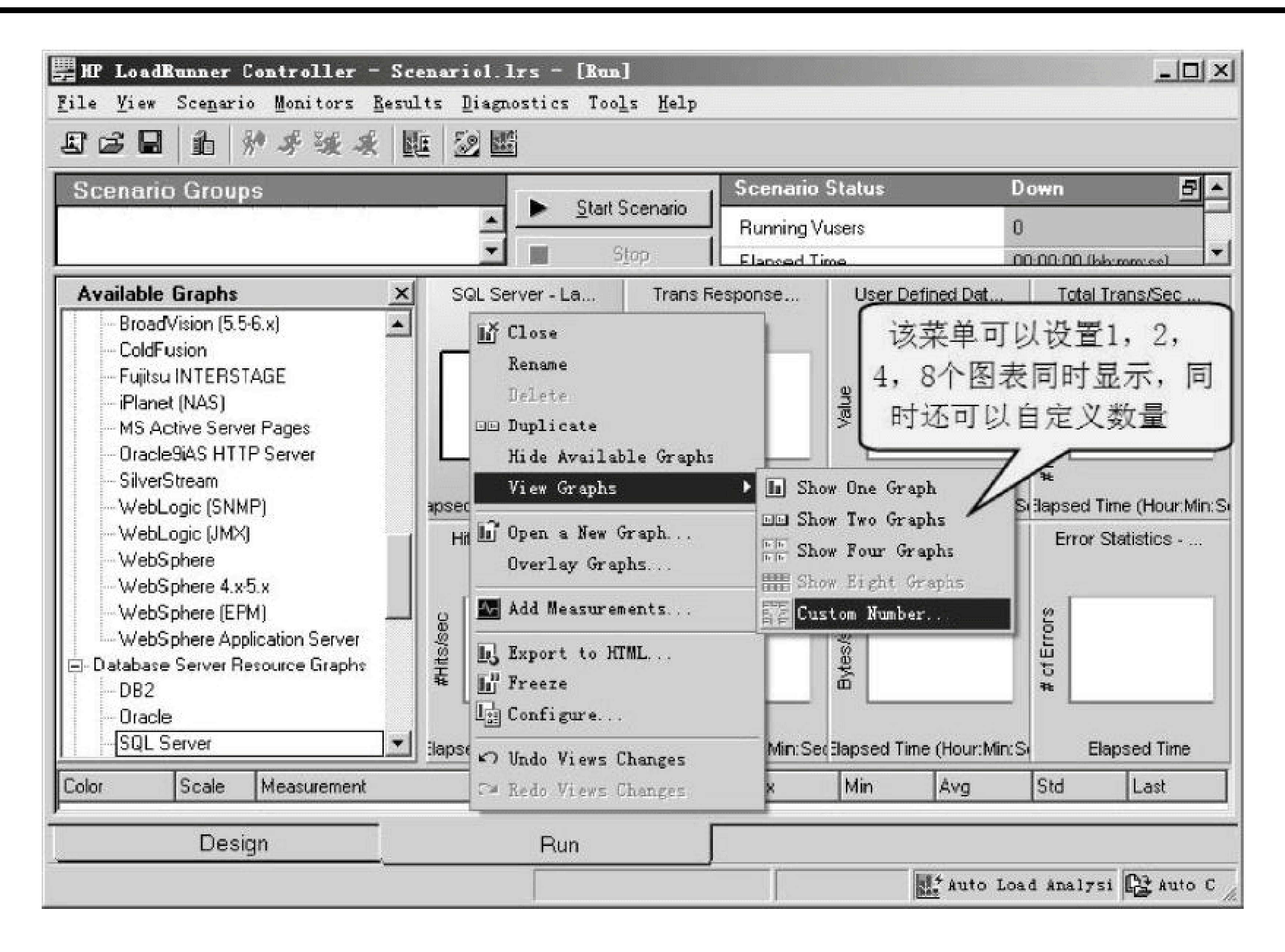


图 11-9　通过菜单设置同时显示多个图表

【监控图表的复制】

在工作中，我们经常会遇到需要显示多个不同计算机的相同性能监控图表。比如，要考察相同 Web 应用程序在不同版本 IIS 服务器中运行的性能情况，就有可能需要多个 IIS 性能图表以监控装有不同 IIS 版本的多台计算机。此时，可以通过复制监控图表来达到目的。在可用图表清单中右击 MS IIS 项，在弹出的快捷菜单中选择 Duplicate 选项，一个新的名为 Copy of MS IIS 的节点将会出现。为了显示方便，可以将其修改为更有意义的名称，再加入到控制器界面中。另外，还要为该监控指定用于对照的机器名称，如图 11-10 所示。

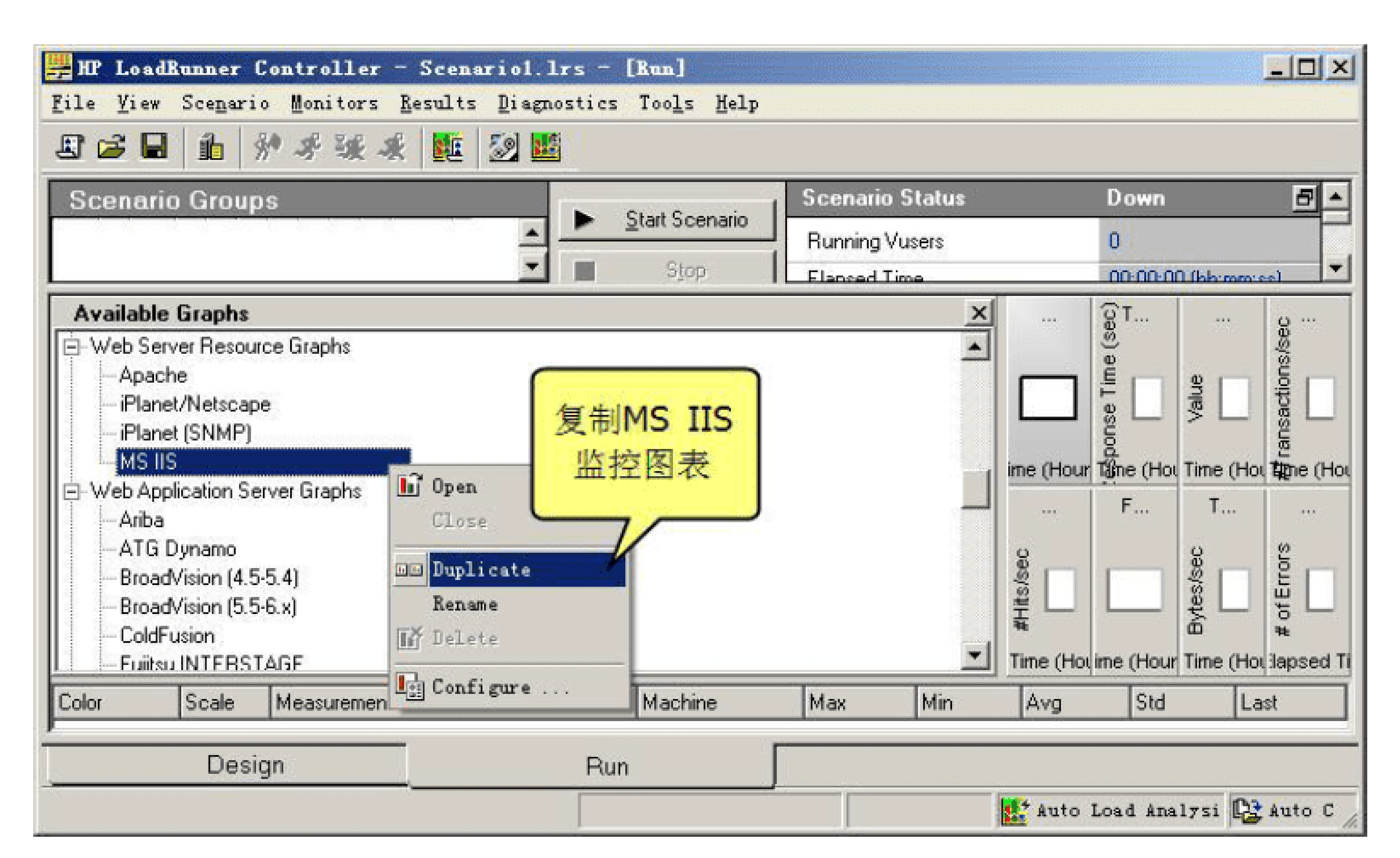


图 11-10　复制一个监控图表

【快速添加监控图表】

快速添加监控图表的方法很简单：直接选中控制器左边可用监控列表中的某项，按住鼠标左键不放，拖动到右边的图表显示区再松开即可。

本节讲述了 LoadRunner 监控图表的创建、设置以及一些技巧。有了监控图表的帮助，

测试工程师才能获得相关性能计数器的数值，得到性能测试的结果，从而为进一步地分析打下坚实的基础。

在 11.2 节，将介绍 LoadRunner 中的函数应用，以完善脚本，为场景执行做准备。

11.2　LoadRunner 中的函数

在本书前几个章节，都或多或少的涉及到了 LoadRunner 中的函数。截至目前，我们知道 LoadRunner 函数可以帮助我们以参数化的形式保存每次登录网站的会话 ID，还可以在场景中设置用户自定义数据点以自由记录当时数据。实际上，LoadRunner 函数所能做的远远不止这些，本节将对其进行专门的介绍。

由于对 LoadRunner 函数的使用需要一定的编程知识，本书将这部分内容放在了后面的内容中，以区分难度，循序渐进。另外，函数有很多，书中只能选取几个为例，介绍应用的过程，读者应该在实际工作中不断使用新的函数，举一反三，从而真正达到熟练的程度。

11.2.1　LoadRunner 函数的简单理解

函数是 LoadRunner 提供给性能测试工程师的利器，有了它，性能测试工程师可以对脚本进行更为自由的开发，更适应实际测试的需求，进一步扩展脚本的功能。

举一个来源于生活的例子，使用 LoadRunner 函数有点类似于在洗印照片店用 Photoshop 后期处理照片的过程：我们作为客户，只需要告诉工作人员要在照片的什么位置进行美化即可。在这个例子里：

- 照片相当于录制下来的 LoadRunner 脚本。
- 工作人员相当于 LoadRunner 的脚本引擎，负责执行我们发出的指令。
- 指令则相当于函数，而且，指令都是工作人员可以完成的，即工作人员可以提供这样的功能。

仔细回想我们发出的指令，一般都类似于这样的形式：请在这里（照片上的某点或区域，可以被称为参数）把脸色（被操作的对象）调浅一点（最终的期望结果，可以被称为返回值）。LoadRunner 函数也是类似的形式，它的格式为：

```
返回值 函数名称（参数列表）;
```

对于 LoadRunner 函数，虽然格式固定，但为了满足不同的使用习惯，具体的写法可以分为 C 语言、Java 语言和 Visual Basic 语言 3 种。用常用的 C 语言方式说明上一个式子，以用户自定义数据点这个函数为例，则是：

```
Int  lr_user_data_point(const char *sample_name, double value);
```

11.2.2　在脚本中应用函数

在脚本中应用函数的方法很简单，请看代码 11-1 所列出的脚本。

代码 11-1　在脚本中应用函数

```
Action()
{
    // [WCSPARAM WCSParam Diff1 43 99280.2428513568fVAzHcHptQVzzzzHDAzt-
    ApDztQf]
    // Parameter {WCSParam Diff1} created by Correlation Studio
//找出会话 ID 并设置为参数的功能
    web reg save param("WCSParam Diff1",
        "LB=userSession value=",
        "RB=>",
        "Ord=1",
        "RelFrameId=1.1",
        "Search=Body",
        "IgnoreRedirections=Yes",
        LAST);
//打开指定网址的功能
    web_url("welcome.pl",
        "URL=http://127.0.0.1:1080/WebTours/welcome.pl?signOff=true",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://127.0.0.1:1080/WebTours/",
        "Snapshot=t1.inf",
        "Mode=HTML",
        LAST);
    lr_think_time(8);
//下面是每隔一秒钟对 CPU 进行监测并输出的功能，持续 100 次
    for (i=0;i<100;i++) {
      measure cpu ( );
      cpu_val=cpu_check();
      lr_user_data_point("cpu", cpu_val);
      sleep(1);
    }
...
```

在代码 11-1 中，我们可以发现使用了大量的函数，列举如下。

- Web_reg_save_param()：能够利用包含在 HTML 页面内的动态变化信息创建参数，在此处代码中创建了表明用户会话 ID 的 WCSParam_Diff1 参数。
- Web_url()：打开其后参数列表中 URL 所指定的网址，在代码中是这个地址：http:// 127.0.0.1:1080/WebTours/welcome.pl?signOff=true。
- Lr_think_time()：为脚本加入思考时间，即用户在思考下一步做什么、未在 HTML 页面上发生任何功能操作的时间。
- Measure_cpu()：代码编写者自己创建的函数，用于增加 CPU 的性能计数器度量。注意它后面的括号内为空，也就是说，参数列表是可以为空的。
- Cpu_check()：对 CPU 进行检查，注意它的返回值被赋值给了 cpu_val 这个变量。
- Lr_user_data_point()：LoadRunner 提供的用户自定义数据点函数，将获取的 cpu_val 值输出到监控图表中。
- Sleep()：脚本休眠一秒钟，等待下个循环。

【自定义函数所带来的好处】

除了 LoadRunner 自带的函数之外，LoadRunner 函数还可以自定义，但要求测试工程师完成函数的具体实现，比如代码中的 Measure_cpu()和 Cpu_check()，具体如何获取 CPU 的信息需要测试工程师完成。这是 LoadRunner 为了扩展功能而设置的。在 11.1 节我们讲

解了很多性能监控图表的使用，但是，有很多产品 LoadRunner 还是不支持的，在这样的情况下，性能测试工程师可以开发自定义的函数，也可以做到获取默认不支持产品的性能指标。

11.2.3　Web 应用常见函数列表

表 11-8 列出了与 Web 应用性能测试可能大量使用的重要函数，它们都处于 Web/HTTP 协议脚本中。对于 LoadRunner 自带函数的命名，一般都采取协议名开头的形式，可以发现，Web/HTTP 特有的函数都以 Web 开头作为前缀。

表 11-8　Web/HTTP协议脚本中的重要函数

函 数 名 称	函 数 说 明
Web_custom_request()	使用 HTTP 支持的任何方法来创建自定义 HTTP 请求
Web_image()	在网页某指定图像上单击鼠标
Web_link()	在网页某指定文本链接上单击鼠标
Web_submit_data()	执行表单提交
Web_submit_form()	执行表单提交
Web_url()	加载指定网址
Web_find()	在网页中搜索指定的文本字符串
Web_global_verification()	在随后所有的 HTTP 请求中搜索文本字符串
Web_image_check()	验证指定的图片是否存在于网页内
Web_reg_find()	在随后的 HTTP 请求中对文本字符串搜索进行注册
Web_create_html_param()	将网页上的动态信息保存为参数
Web_create_html_param_ex()	将包含在网页内的动态信息创建为参数（使用边界）
Web_reg_save_param()	将包含在网页内的动态信息创建为参数（不使用边界）
Web_set_max_html_param_len()	设置可以设置为参数的 HTML 字符串的最大长度

除此之外，还有几个不是 Web/HTTP 协议脚本所特有，但对于 Web 应用也非常重要的函数，它们都属于脚本的控制流程函数，如表 11-9 所示。

表 11-9　一些重要的脚本控制流程函数

函 数 名 称	函 数 说 明
Lr_start_transaction()	标记事务的开始
Lr_end_transaction()	标记事务的结束
Lr_rendezvous()	设置集合点以创建虚拟用户（VUser）并发策略
Lr_think_time()	暂停脚本执行，模拟真实用户的思考时间
Lr_user_data_point()	用户自定义数据点函数，将获取的 cpu_val 值输出到监控图表中

11.2.4　学习使用 LoadRunner 函数的方法

LoadRunner 函数是使用 LoadRunner 进行性能测试中较为高级的一部分，由于本书面向初学者，如果深入地探讨恐怕几个章节都无法讲解完毕。进一步学习 LoadRunner 函数，

需要如下的能力：

英文阅读能力：绝大多数较详细的 LoadRunner 函数使用技巧都是英文文档，比如官方的 HP Online Function Reference 等。

编写代码能力与计算机科学知识：这同样也是很难在短期提高的能力。LoadRunner 函数理解和调用并不难，但要在实际工作中灵活运用并且编写自定义函数，还需要对于被测试软件、编程语言、操作系统等拥有较为深入的理解。

以上两点都不是一本书所能介绍完的，可以说，LoadRunner 函数的应用体现了性能测试工程师的综合实力。那么，如何更好地学习和掌握函数呢？读者不妨采取下面两个方法：

- 多录制脚本，分析脚本。录制的脚本中已经包含了大量的 LoadRunner 函数，读者完全可以通过读代码、运行代码、调整修改代码的方式快速熟悉。
- 多查阅惠普的在线函数帮助（Online Function Reference）。

在线函数帮助可以通过依次选择“开始|运行|LoadRunner|Documentation|虚拟用户函数帮助”来打开，也可以在虚拟用户生成器（VuGen）的界面中，依次选择 Help|Function Reference（帮助|函数帮助）命令来打开，如图 11-11 所示。

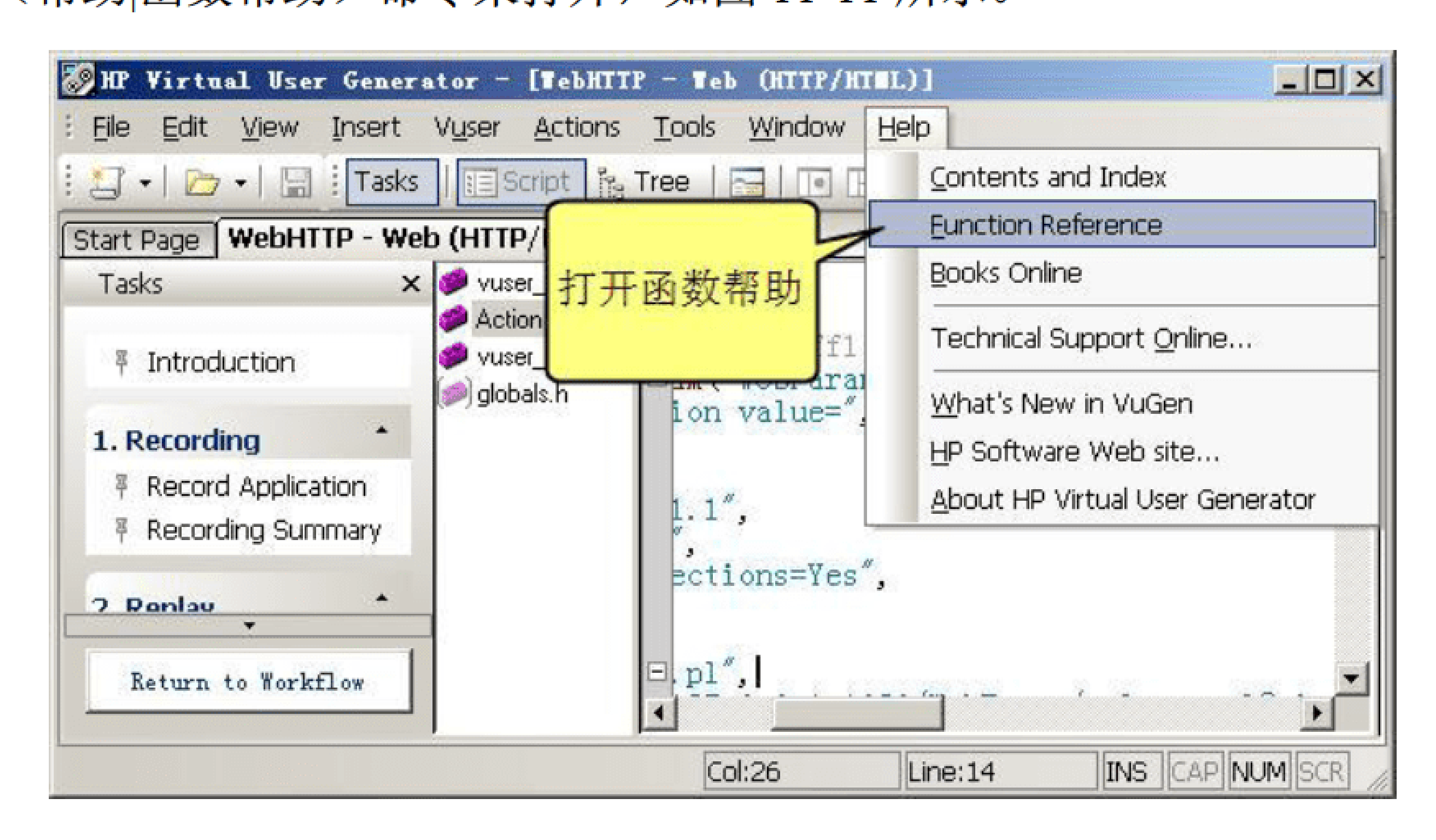


图 11-11　打开 LoadRunner 函数帮助文档

在帮助文档中按照脚本协议详细列举了所有的 LoadRunner 自带函数的格式、使用与代码范例，这是学习与掌握函数使用不可多得的材料。

11.3　本 章 小 结

本章介绍了在性能测试执行之前，对于场景和脚本这两部分内容还需要做的完善工作：为场景添加监控与为脚本添加函数。

监控图表是观察 Web 应用性能表现的窗口。通过增加各种类型的监控图表（比如网络服务器、数据库等），性能测试工程师可以对被测试软件的性能有直观具体的认识，同时，也为分析结果提供了大量的原始数据。为场景添加监控是必须进行的，主要步骤如下：

（1）配置被监控软件的客户端（对于有些监控则不必要）

（2）添加各类型的监控图表。

（3）添加被监控计算机以及度量性能计数器。

如果被测试的产品不在 LoadRunner 所支持的默认监控之列，则需要自定义实现监控功能。在这样的场景下，监控需要和 LoadRunner 函数相结合。

所谓 LoadRunner 函数，是 LoadRunner 提供的一个强大的功能接口。函数包括默认提供和自定义函数两种，用于实现与扩展脚本功能。函数的命名一般以脚本所使用协议作为前缀，如果多个协议共有，则以 LR 作为前缀。

LoadRunner 函数的灵活使用体现了性能测试工程师的综合实力，需要在实际工作中不断学习，不断提高。

在第 12 章，我们将执行设置好的场景与脚本，并对结果进行分析。

第 12 章　执 行 场 景

本章将进入性能测试的实施阶段：执行场景。对于性能测试的执行，LoadRunner 需要通过控制器（Controller）组件来实现；执行完毕后，LoadRunner 将自动打开分析器，显示本次执行的测试分析概要。

由于在 LoadRunner 中，根据默认设置，执行场景与显示分析概要是一气呵成的，另外服务质量协议也需要在控制器和分析器中结合起来学习，因此本章对这两部分一起进行介绍。本章还会对简单介绍测试结果图表，为下一章通过图表分析性能问题打下一定基础。

12.1　LoadRunner 性能测试的执行

LoadRunner 执行性能测试实际上就是执行前文中已经设计好的场景。但是，执行的方式、计划等方面需要额外设置。在前面的章节中读者已经对控制器的界面非常熟悉，本章中几乎所有有关场景执行的控制都处于 Run（运行）选项卡中。

12.1.1　执行性能测试

运行 LoadRunner 的控制器（Controller），单击“文件|打开”选择预先保存好的场景，再单击控制器下方 Run（运行）标签，即可在如图 12-1 所示的界面中进行场景执行的设置与控制。在窗体中上部分，Start Scenario（开始场景）按钮负责启动性能测试的运行；当场景处于运行过程中，可以单击 Stop（停止）按钮结束当前的性能测试。

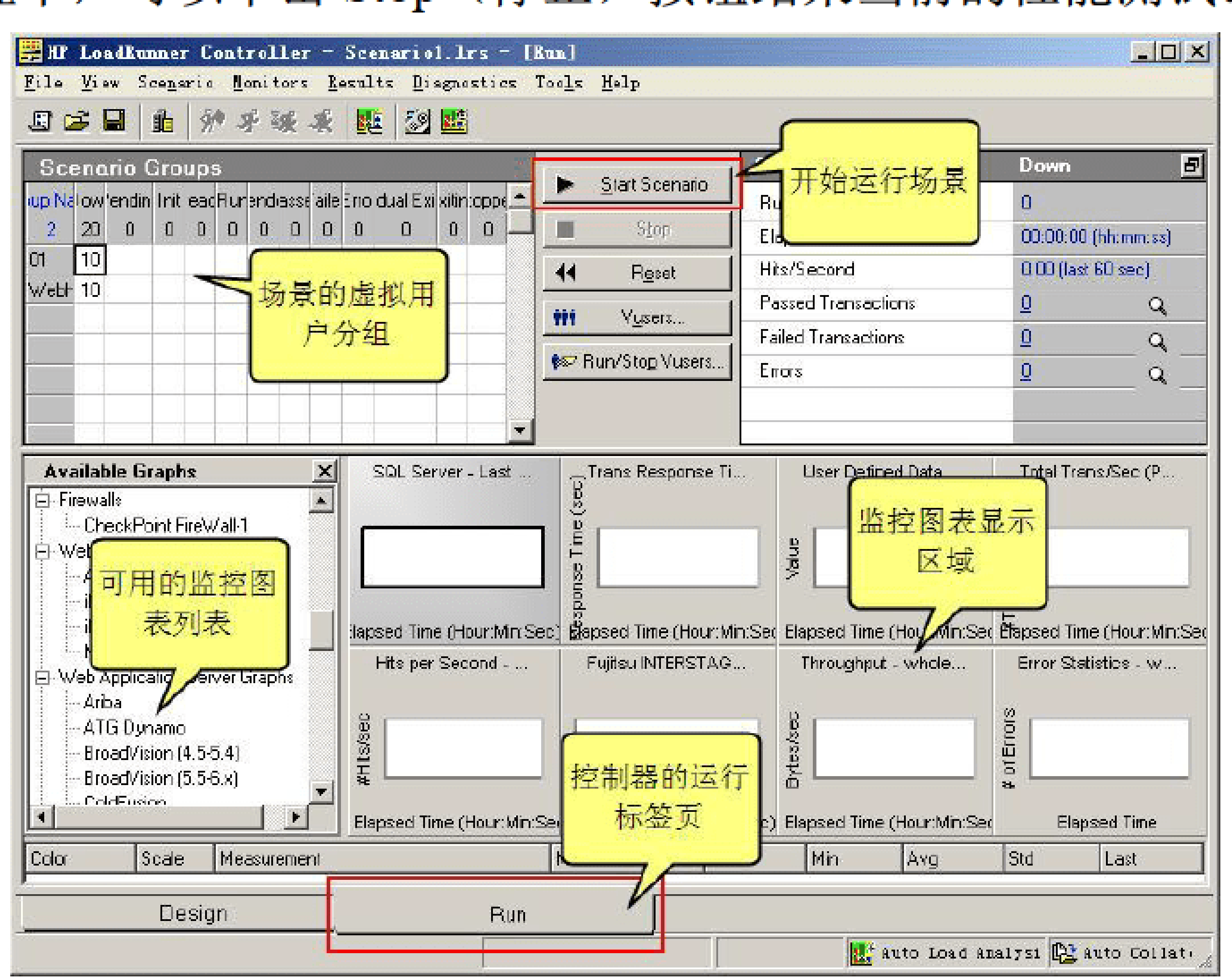


图 12-1　执行性能测试的控制按钮

【性能测试结果的保存】

性能测试最终会产生一些结果，之后测试工程师再对这些结果进行分析，形成测试报告。因此，在执行测试之前，要指定测试结果存放的位置。LoadRunner 会在我们单击开始场景之后立即弹出如图 12-2 所示的对话框，根据实际情况进行设置即可。单击 OK 按钮，性能测试的执行就正式开始了。

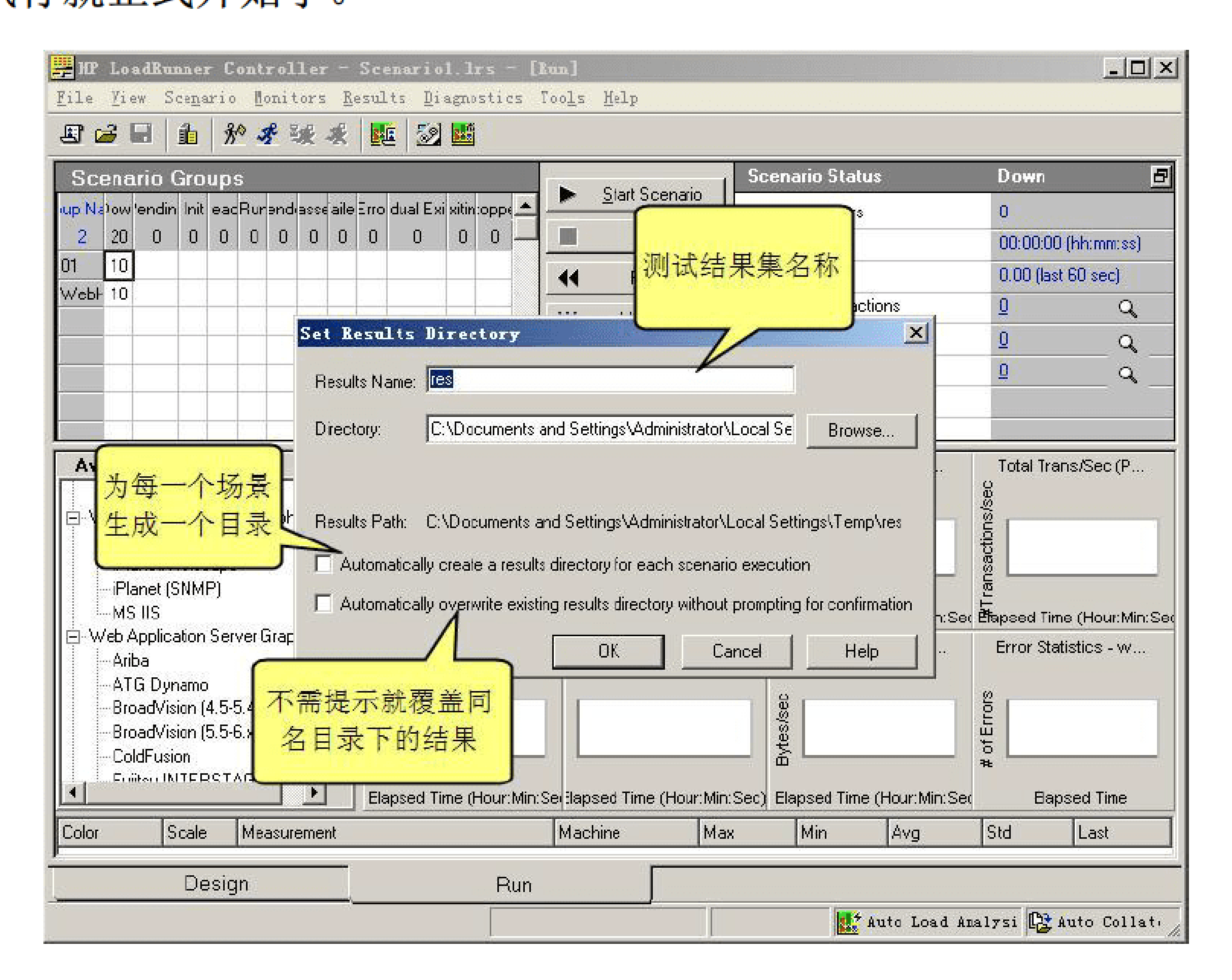


图 12-2　为性能测试指定结果数据存放位置

12.1.2　场景执行时的控制器

一旦场景开始运行，LoadRunner 控制器界面会实时显示已经加入的各个监控图表的数据变化曲线。

除此之外，控制器还提供了更多的信息。在控制器窗体左上方的虚拟用户组视图中有详细的表格，其中每一行代表场景中的一个虚拟用户组，每一列则代表了虚拟用户组中各种状态的虚拟用户数量，如图 12-3 所示。在场景运行期间，我们可以发现图 12-3 中各状态下的数量是不断变化的，最终所有的虚拟用户都将处于运行状态。

当单击结束场景或者场景运行时间结束之时，虚拟用户的状态将从运行转为逐渐退出、退出以至完全停止状态。

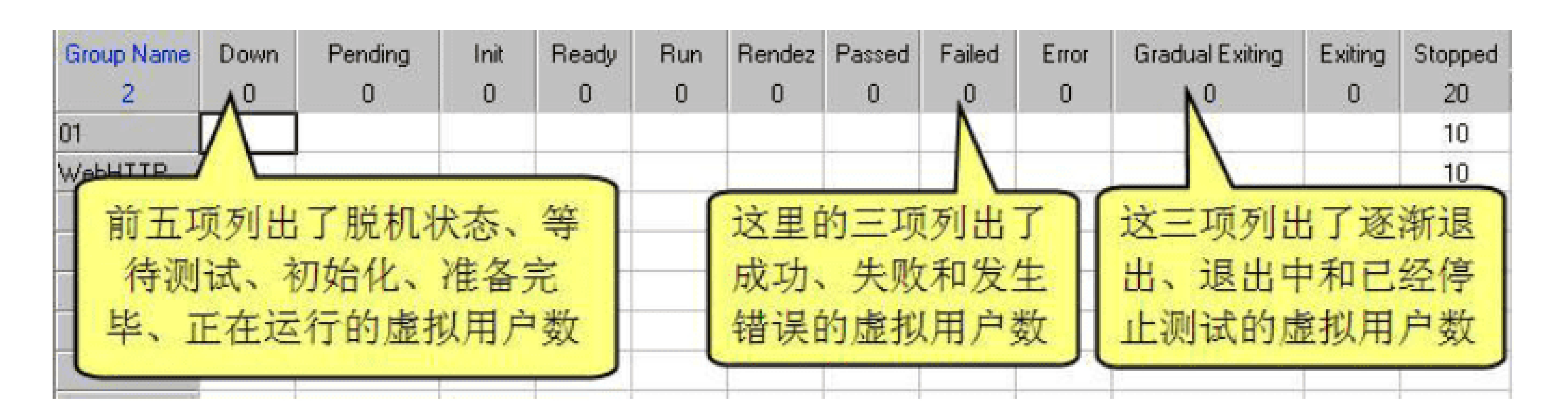


图 12-3　虚拟用户组能够显示处于各种状态下虚拟用户的数量

虚拟用户的这几种状态可以用表 12-1 来说明。

表 12-1　虚拟用户在场景执行期间可能的状态

状　　态	说　　明
关闭（Down）	虚拟用户尚未准备运行
待命（Pending）	虚拟用户待命，等待可用的压力生成器调用
初始化（Init）	虚拟用户在指定的压力生成器上进行运行初始化脚本
就绪（Ready）	虚拟用户初始化脚本执行结束，准备运行
运行（Run）	虚拟用户在压力生成器上运行指定脚本
集合（Rendez）	虚拟用户到达集合点，等待条件满足后 LoadRunner 的释放，进行脚本的下一步
成功（Passed）	虚拟用户结束运行，脚本成功通过
失败（Failed）	虚拟用户结束运行，脚本失败
错误（Error）	虚拟用户发生错误
逐步退出（Gradual Exiting）	虚拟用户正在完成退出前的脚本指定操作
正在退出（Exiting）	虚拟用户正在退出
已停止（Stopped）	停止按钮按下后，虚拟用户停止工作

图 12-4 显示了某一次性能测试执行阶段过程中，控制器的界面。可以发现，界面分为上下两个部分，分别代表了状态和性能指标两类信息。

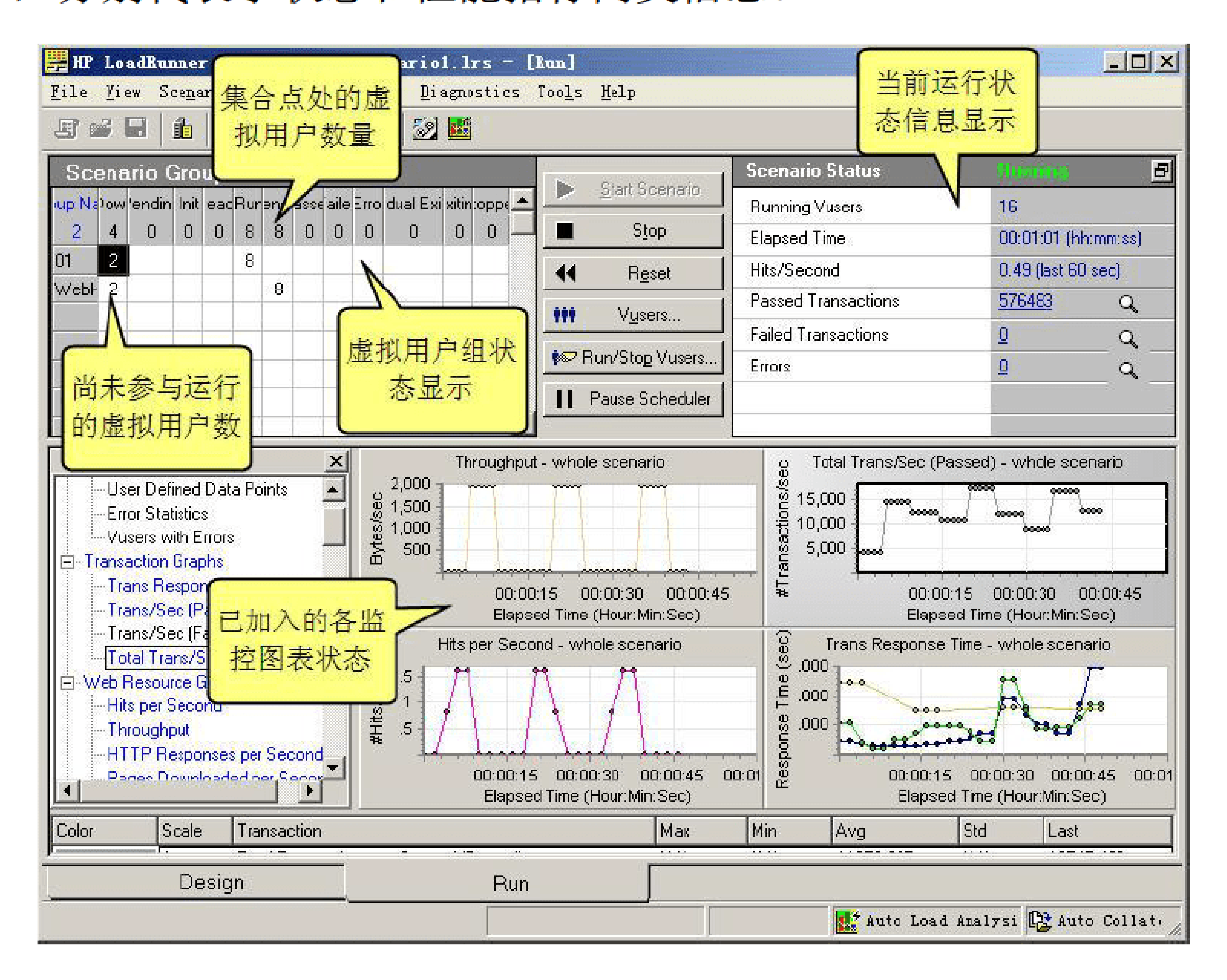


图 12-4　执行性能测试时控制器的界面示意

12.1.3　场景执行过程中的状态信息

在图 12-4 的右上角，显示有场景运行状态信息，包括如下几个常用度量的状态，便于运行时查看，如图 12-5 所示。

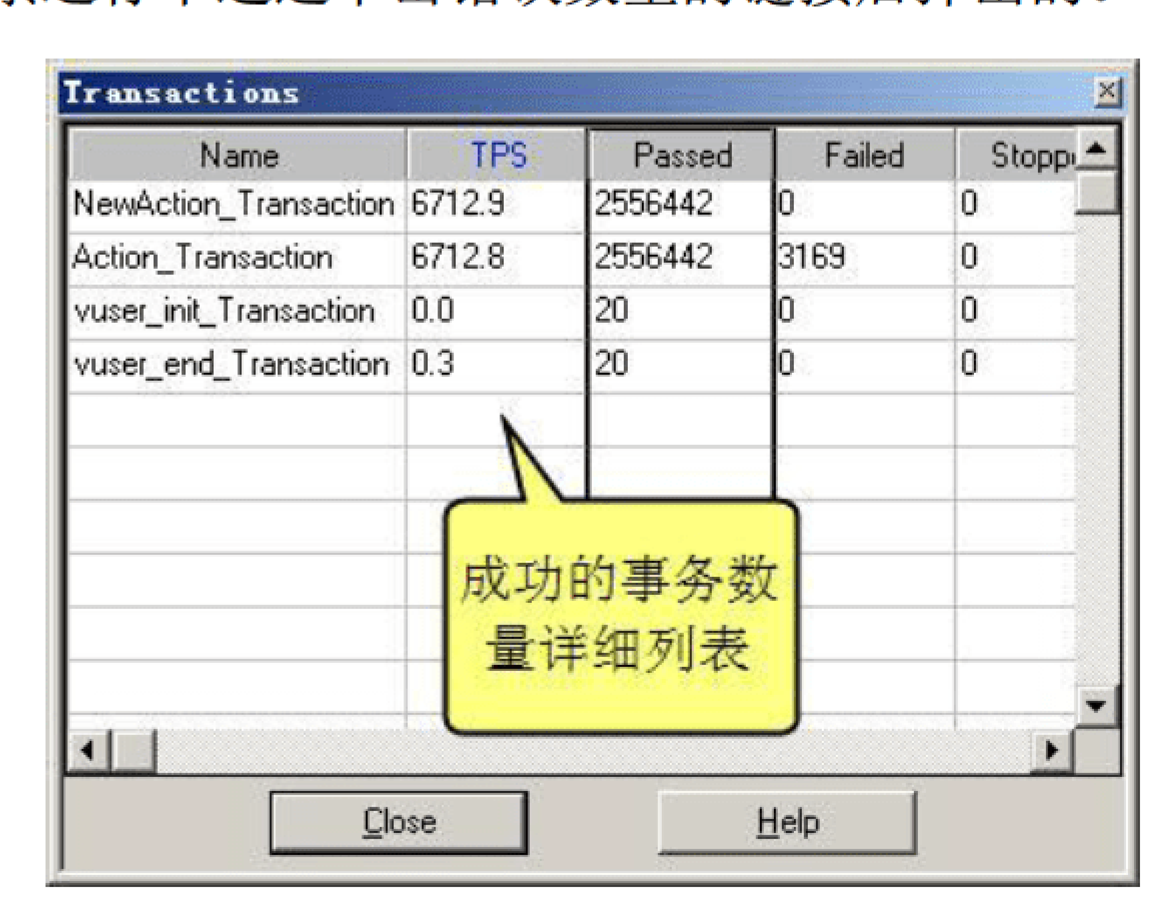
图 12-5 中列出的常用度量为：

- 场景状态（Scenario Status）：正在运行（Running）用绿色表示。
- 运行中的虚拟用户（Running Vusers）：当前正在执行脚本的虚拟用户数量。
- 已用时间（Elapsed Time）：用小时：分钟：秒的形式呈现。
- 点击次数每秒（Hits/s）：显示的是截至目前最后一分钟内的平均值。
- 成功的事务数量（Passed Transactions）：当前场景运行结果中所有成功事务的数量。
- 失败的事务数量（Failed Transactions）：当前场景运行结果中所有失败事务的数量。
- 错误数量（Errors）：当前场景运行结果中所有错误数量。

Scenario Status	Down
Running Vusers	0
Elapsed Time	00:06:33 (hh:mm:ss)
Hits/Second	0.00 (last 60 sec)
Passed Transactions	5112924
Failed Transactions	3169
Errors	9507

图 12-5　场景运行状态信息视图

其中最后 3 项数值都会紧跟一个放大镜的图标，单击该图标按钮可以获得更加详细的显示，如图 12-6 所显示为成功事务数量详情图。失败事务数量与错误数量均用红色背景标出以示醒目，它们的数值也是链接，可以单击以查看更多的内容。图 12-7 所列出的错误列表窗口，正是某次场景运行中通过单击错误数量的链接后弹出的。


Transactions

Name	TPS	Passed	Failed	Stopp
NewAction_Transaction	6712.9	2556442	0	0
Action_Transaction	6712.8	2556442	3169	0
vuser_init_Transaction	0.0	20	0	0
vuser_end_Transaction	0.3	20	0	0

图 12-6　成功事务数量详细列表

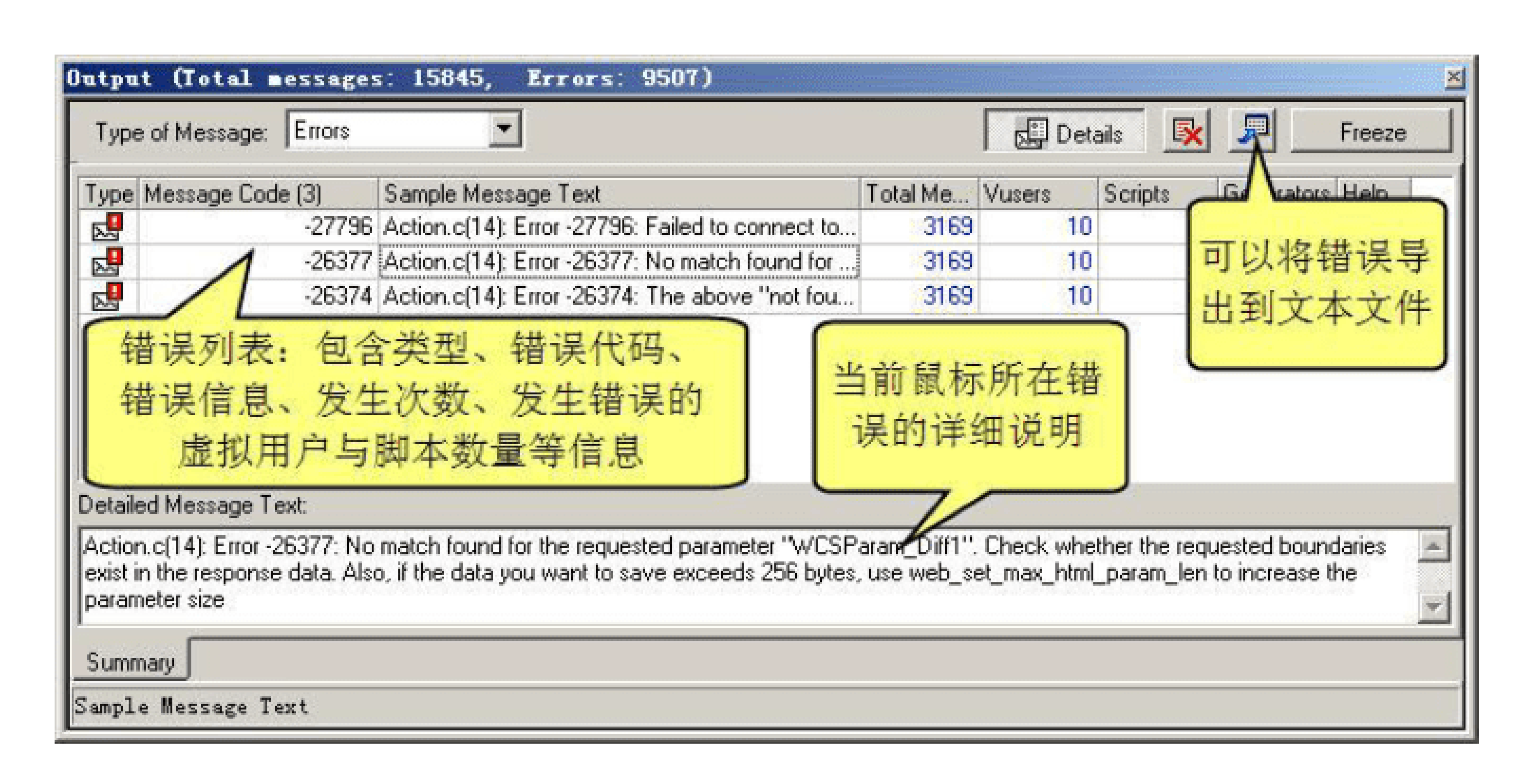


图 12-7　错误列表窗口

【查看详细错误】

如果鼠标悬停在错误列表窗口的某行错误之上，那么错误的详细说明将在图 12-7 窗体下方的文本窗口中显示出来。从图 12-7 中可以发现，错误发生在脚本文件 Action.c 中，具体信息是找不到名称是 WCSParam_Diff1 的参数，这是由于在本例中被测试 Web 应用服务没有正常启动的缘故，该参数并没有生成。

另外，单击图 12-7 中每个包含链接（下划线作为链接提示，与网页类似）的数值，都会增加一个新的名为过滤器（Filter）的标签页，以显示当前选择项目下具体的错误信息。如图 12-8 是单击图 12-7 某 Vusers 列内数值得到的过滤器结果，单击其他列还将得到其他有用的错误信息。

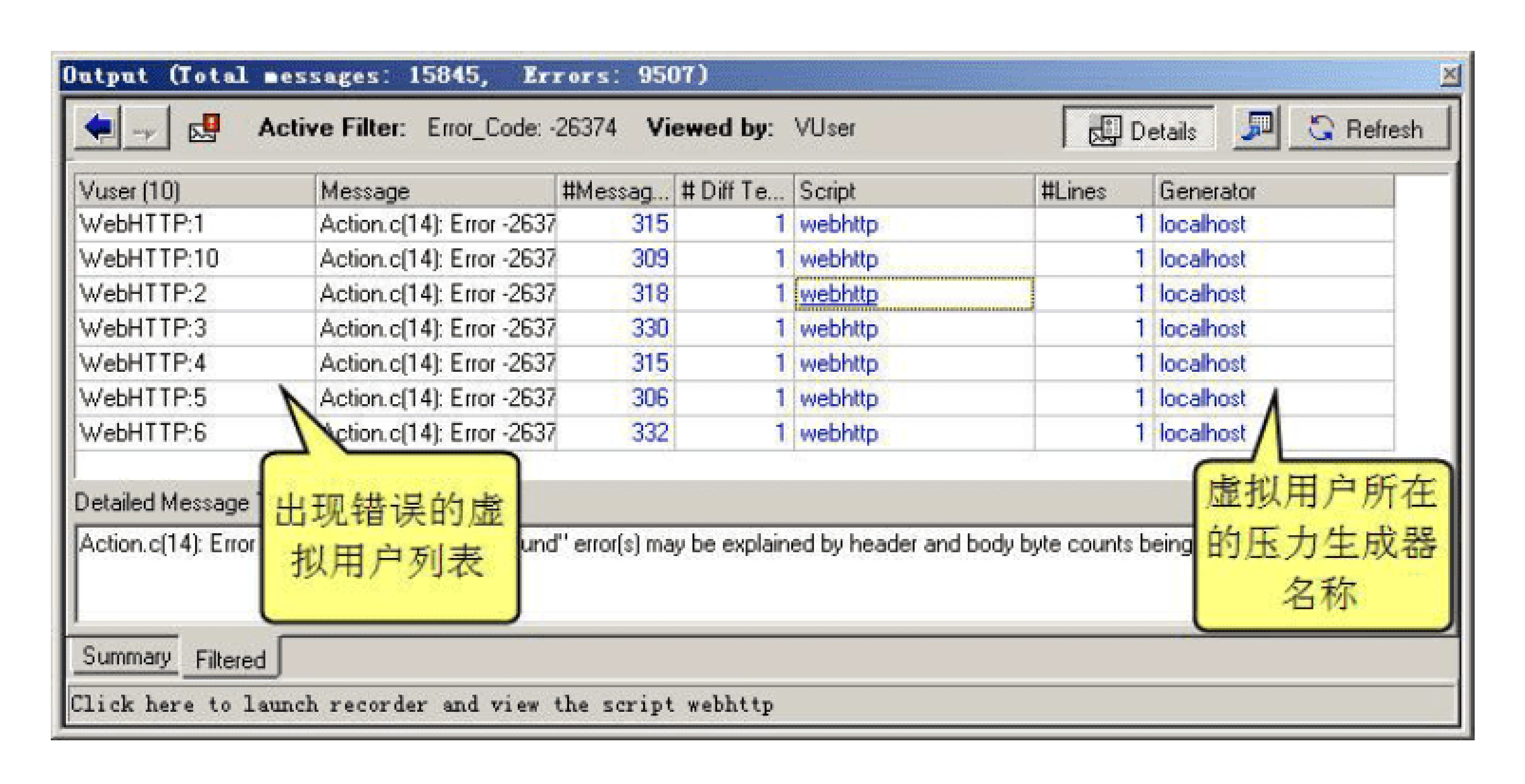


图 12-8　出现错误的虚拟用户列表

通过查看错误窗口，可以发现当前场景、脚本的一些问题，便于对性能测试进行完善。

12.1.4　场景执行完毕

当场景执行完毕时，LoadRunner 默认会自动打开分析器（Analysis），经过短暂的报告生成过程，最终形成测试结果，如图 12-9 所示。测试结果包括测试总结报告、测试会话分类浏览、属性窗口等几个部分。

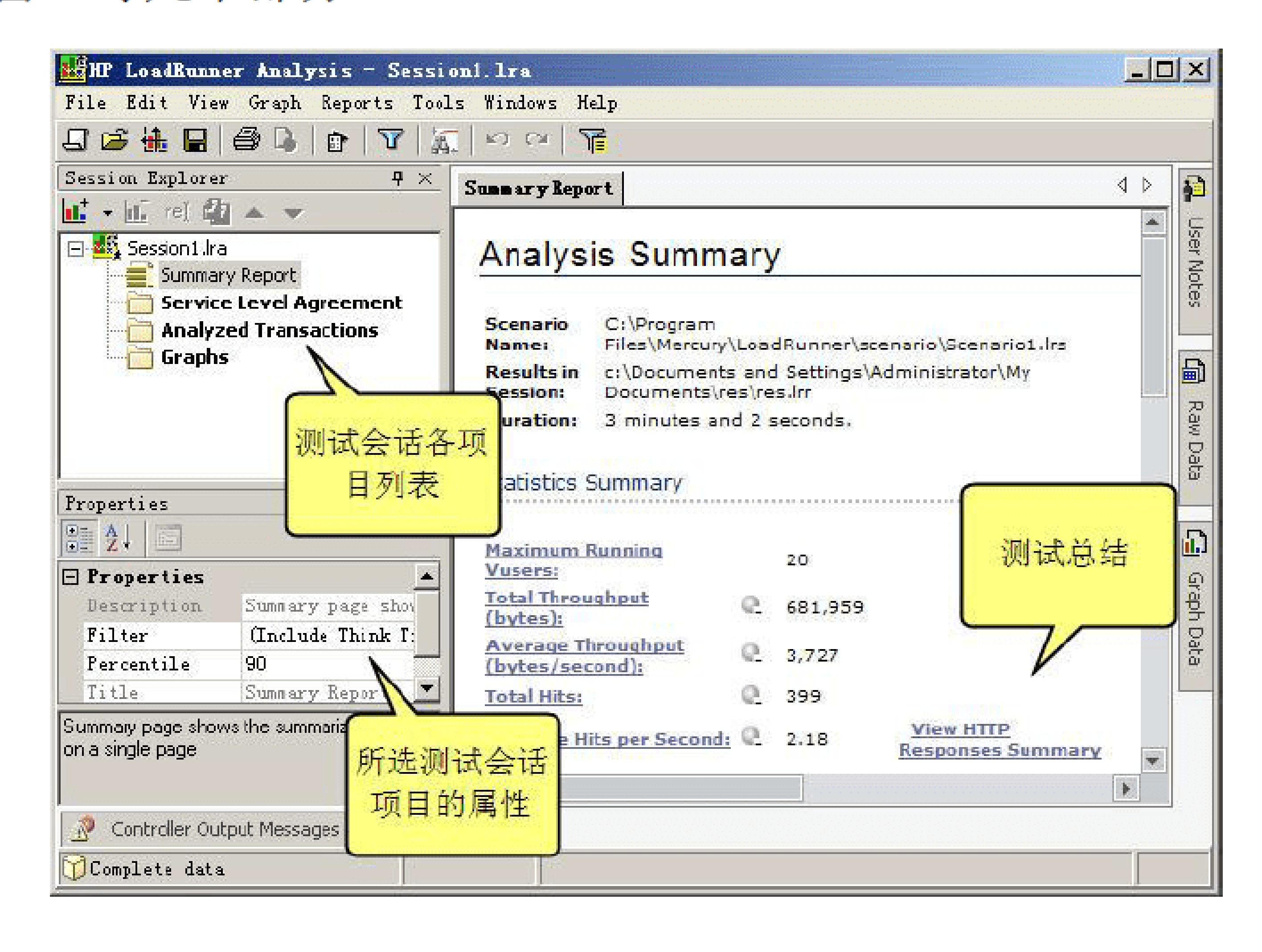


图 12-9　测试执行完毕后自动打开分析器显示性能测试报告

【自动打开分析器的切换】

如果没有选择控制器菜单中的 Results|Auto Load Analysis（结果|自动打开分析器），则上述结果不会自动产生。LoadRunner 之所以包含这样的设计，是为了更加灵活，使其适用于性能测试场景的初始调试排错阶段。

对于测试后的结果，还可以进行其他几个设置，比如是否自动合并各性能监控表（将在后面的小节提到）等，它们都处于控制器的 Result（结果）菜单下。该菜单的所有选项如图 12-10 所示。选择 Result Settings（结果设置）菜单，LoadRunner 将会弹出之前提到过的设置结果文件存放位置窗体，如图 12-2 所示。

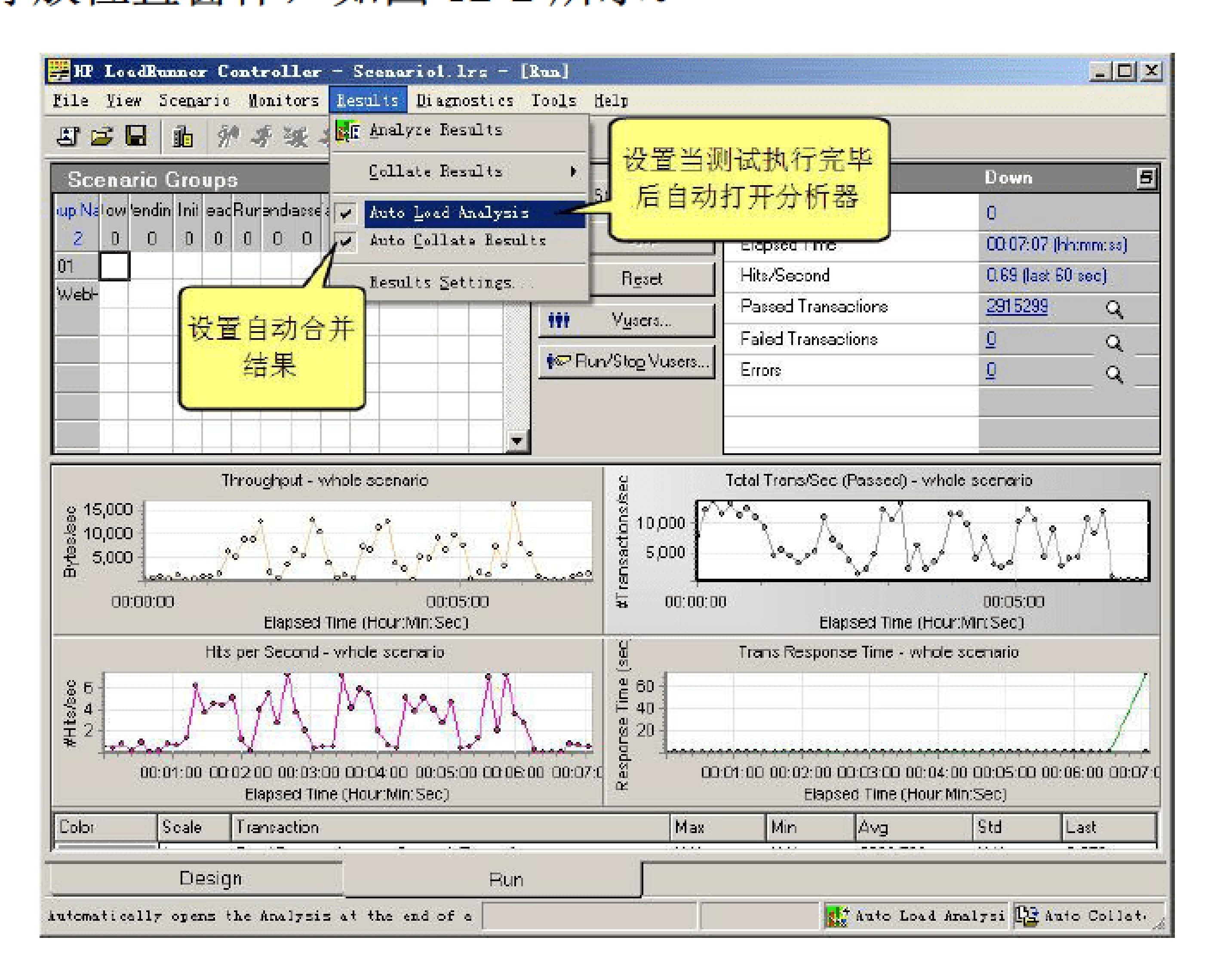


图 12-10 有关测试结果的几个设置

熟练掌握分析器的使用，与熟悉 SLA 的知识是分不开的。在 12.2 节，我们需要进一步了解第 11 章提到过的服务质量协议（SLA）相关背景与设置。场景中有无 SLA，会对分析器的结果报告产生影响；并且只有在熟悉 SLA 之后，测试工程师才会更好地理解分析器提供的概要报告，从而发现可能的性能问题。

12.2 服务质量协议（SLA）

服务质量协议顾名思义就是两方对于服务质量所达成的一致文本。在 LoadRunner 之中，简单地说，它指的是一系列预设的性能期望值，一般可以在性能测试计划或者性能测试目标文档中找到。在前面的章节中提到过基线（Baseline），服务质量协议就相当于这样的基线，如果某个性能指标超过了 SLA 所规定的数值，则说明此项指标上服务质量（即性能）出现了无法达到的情况，需要优化或者根据情况调整 SLA 本身。

12.2.1 添加服务质量协议（SLA）

SLA 是 LoadRunner 9 新增添的功能。设置 SLA 的方法主要通过控制器主界面右上方

的服务质量协议窗口来实现，如图 12-11 所示。



图 12-11　服务质量协议窗口界面

单击 New 按钮，控制器弹出 SLA 配置向导的说明页面，单击 Next 按钮后将进入服务质量协议向导的 Goal Definition（目标设置）窗体部分，如图 12-12 所示。



图 12-12　服务质量协议的目标类型设置

需要注意的是，图 12-12 中列出了两种 SLA 状态决定方式，只能选择它们之中的一种：

- 第一个选项将使得 SLA 与性能测试的执行时间相关，简单地说，关注于每秒的事务（Transaction）数量和每秒发生的错误，粒度更为细致。
- 第二个选项使得 SLA 状态被性能测试的执行所决定，关注总体指标，比如该选项下供单项选择的具体指标，都是以每次运行为单位的。

【SLA 状态】

所谓 SLA 状态，就是指是否满足服务质量协议，即通过（Pass）或者未通过（Fail），当然还有第 3 种状态，即无数据（No data），这是针对没有设置 SLA 而言的。

12.2.2　选择时间决定的 SLA

在图 12-12 中选择第一个单选框，使得 SLA 状态被时间线上的某时间间隔所决定，即每隔一段时间间隔获取数据考察 SLA 是否满足的方式。该选项包含 2 项指标：

- 每间隔时间的平均交易响应时间（Average Transaction Response Time per time interval）；

- 每间隔时间的每秒错误数量（Errors per second per time interval）。

顾名思义，前者监视每一段时间内平均交易响应时间数值的变化，后者监视每秒发生的错误数量。那么，每间隔时间在哪里体现呢？这将在 12.2.4 节中介绍。

为举例方便，本书选择窗体单选框第一个子选项：每间隔时间的平均交易响应时间（Average Transaction Response Time per time interval）进行介绍。

（1）选择后单击“下一步”按钮，弹出界面如图 12-13 所示，该步骤的目的是为了场景中的一个或多个事务添加 SLA。

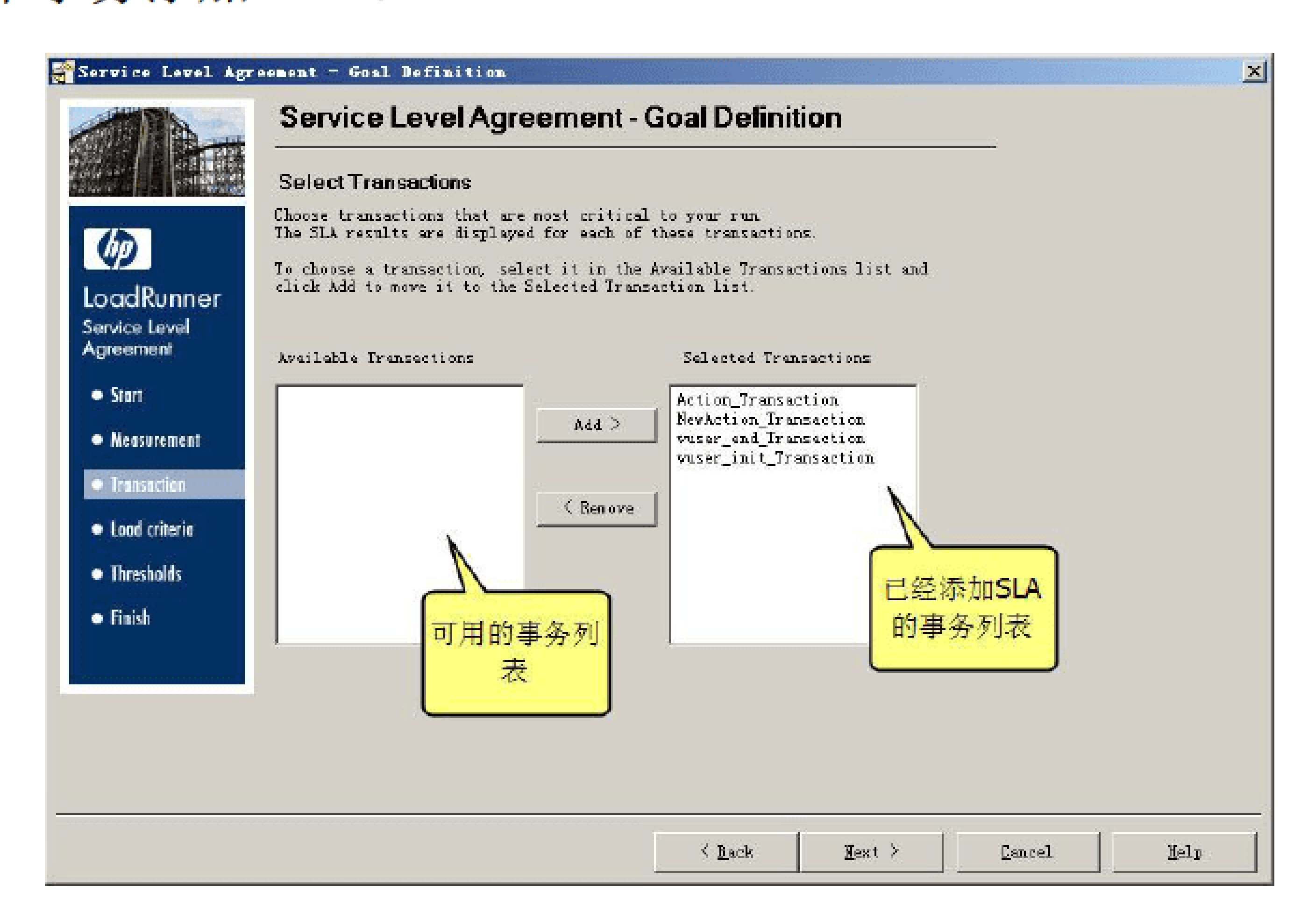


图 12-13　从事务列表中选择以增加 SLA

（2）在图 12-13 中单击 Add 按钮 4 次，将可用事务中的各项目全部添加，单击 Next 按钮，如图 12-14 所示。该步骤的目的在于为当前被设置的 SLA 添加工作负荷并设置工作负荷的具体数值。本书举例选择了运行用户数量为工作负荷的产生标准，并且选择了大于等于 10 作为具体数值。设置具体数值是为了更加灵活：可以不把场景用户组中所有虚拟用户的行为作为判断是否满足 SLA 的标准。



图 12-14　为被设置的 SLA 添加工作负荷产生标准

（3）单击 Next 按钮，在图 12-14 中为图 12-13 所选中的每一个事务单独设置 SLA 的通过标准，这个标准也可以叫做阈值（Threshold）。

当然，对于所有事务，完全能够使用相同的标准，可以在图 12-15 中窗体下方对此进行设置并单击 Apply to all transactions（应用到所有事务）按钮以生效。在场景的实际运行过程当中，如果实际值超出这些设置好的数值，SLA 状态将被标记为未通过，即未达到服务质量要求。

（4）在图 12-15 中逐一或者统一设置各项数值完毕后，单击 Next 按钮，完成对当前 SLA 的定义，如图 12-16 所示。

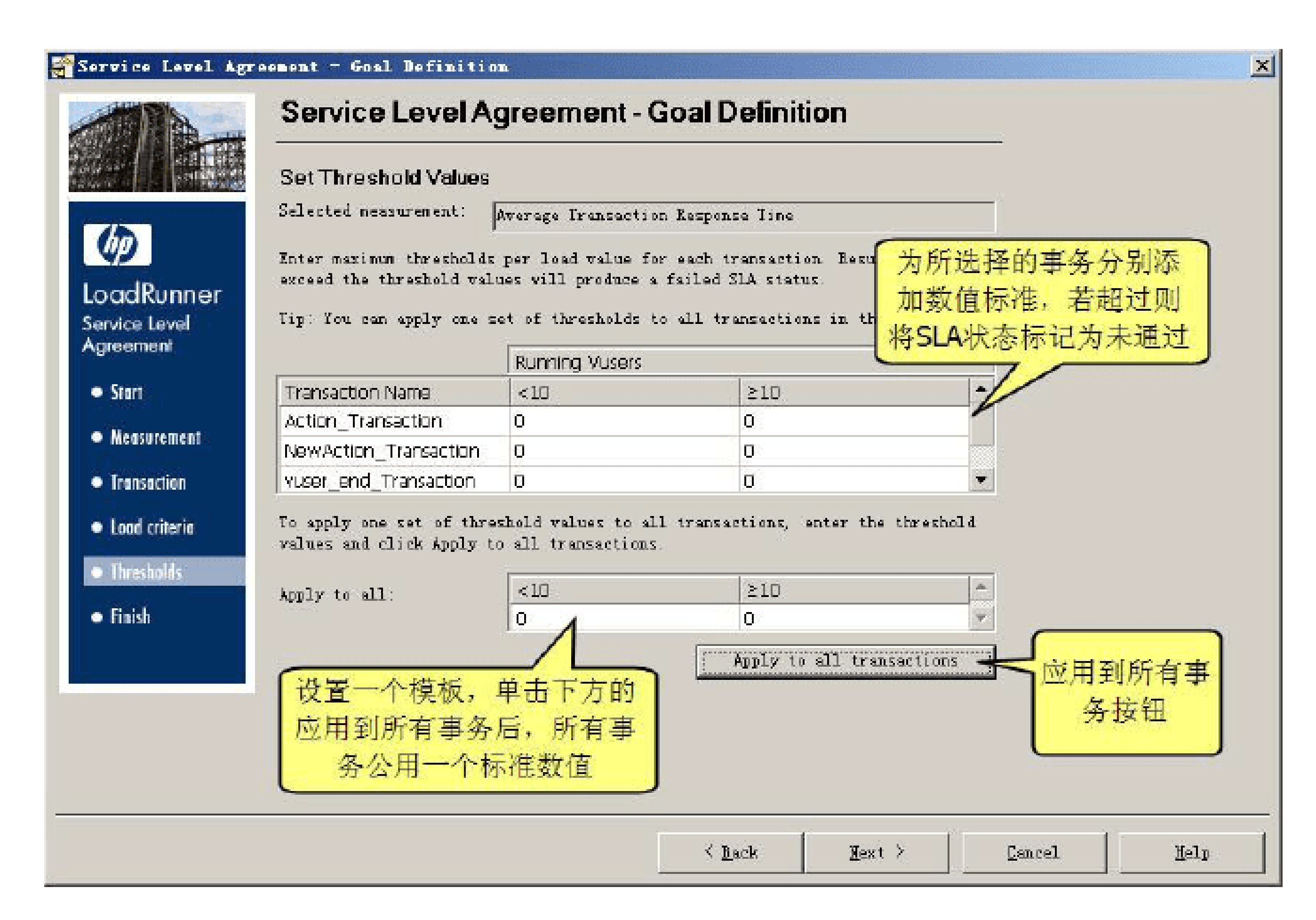


图 12-15　为事务设置 SLA 标准阈值

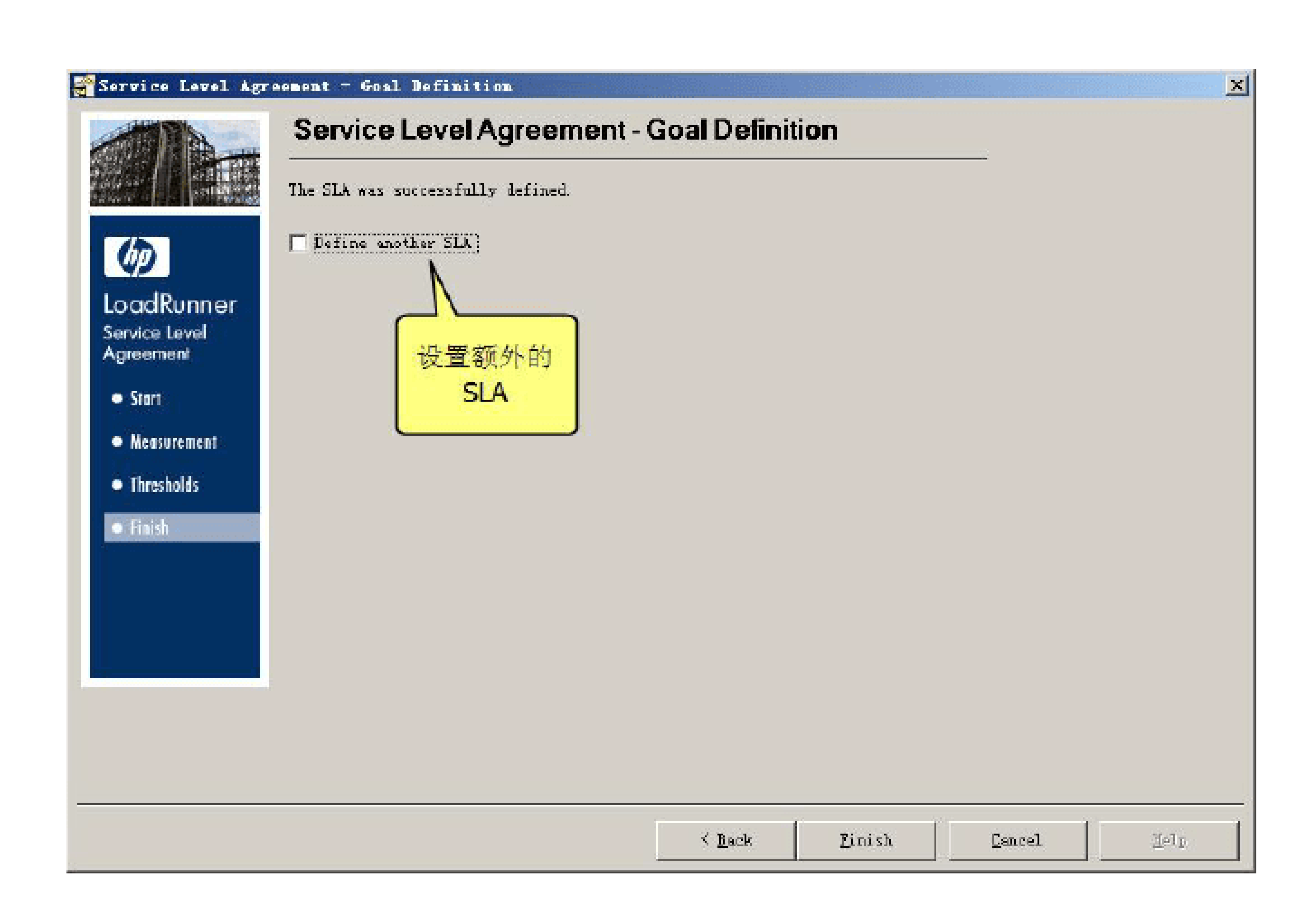


图 12-16　结束当前 SLA 的设置

12.2.3　选择运行决定的 SLA

在图 12-12 中选择第 2 个单选框，使得 SLA 状态被每次运行所决定。在选项下面还有 4 个子选项，分别对应如下 4 个度量值：

- 每次运行总点击量（Total Hits per run）；
- 每次运行平均每秒点击量（Average Hits/per second per run）；
- 每次运行总吞吐量（Total throughput per run，单位 bytes，字节数）；

- 每次运行平均每秒吞吐量（Average throughput per run，单位每秒字节数）。

（1）为举例方便，本书选择第一个子选项：Total Hits per run（每次运行总点击量）进行介绍。选择后单击 Next 按钮，向导界面如图 12-17 所示。

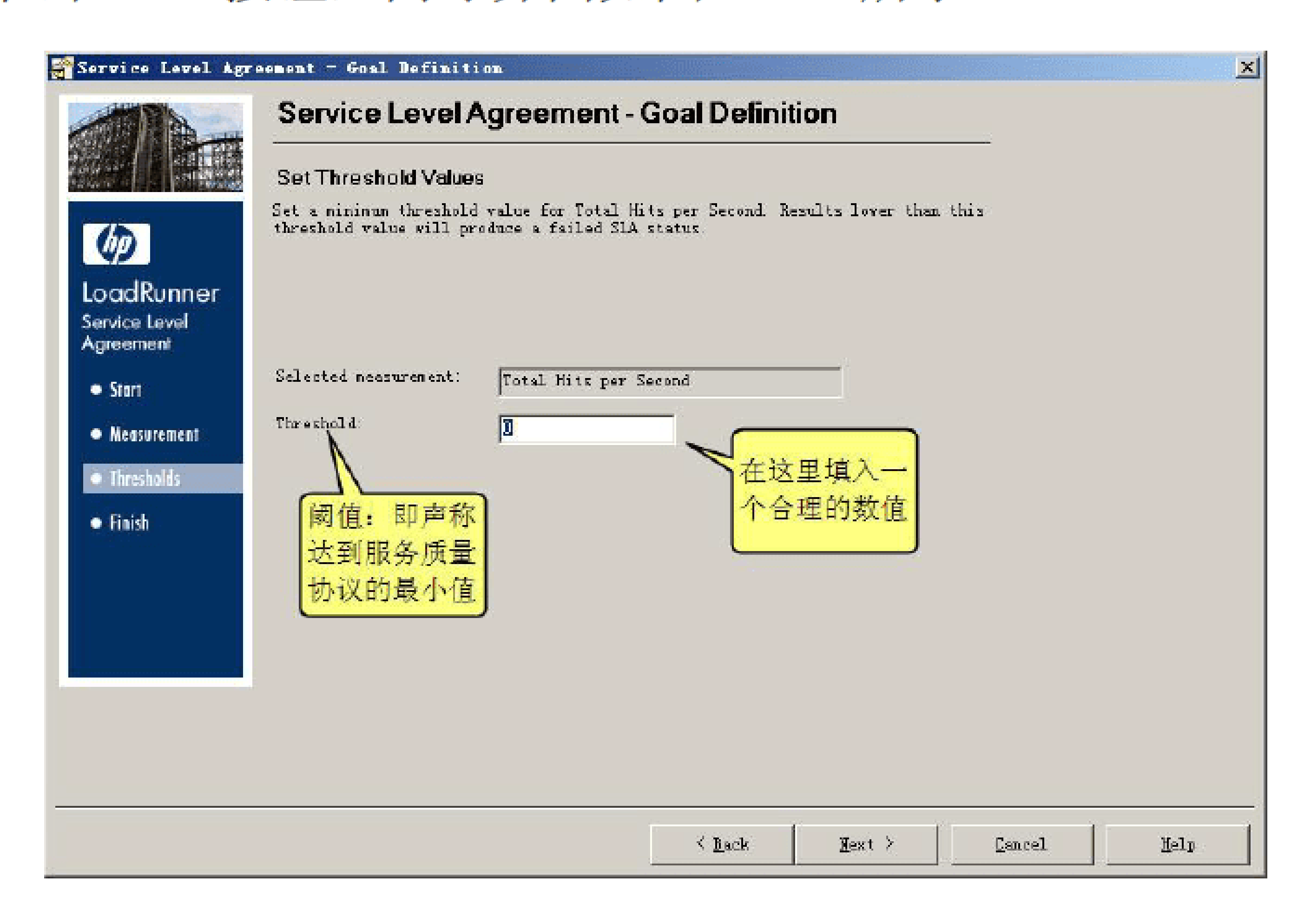


图 12-17　对所选择服务质量协议的阈值进行设置

（2）在图 12-17 中，需要对当前选择的指标设置一个阈值，即声称该项服务质量协议通过的最小值。输入一个合理的数值后，单击 Next 按钮，继续单击 Finish 按钮，即可完成当前 SLA 的设置，如图 12-18 所示。



图 12-18　SLA 设置完成界面

【哪些是通用的设置】

需要注意的是，在图 12-12 中所选择的类型和子度量值不同，后续的设置界面可能不同，但都不会脱离对阈值，即性能测试目标值的设置。读者完全可以举一反三，在实际工作中很快熟悉。

如果需要，可以选中图 12-18 中设置额外 SLA 的选项，将重复之前的步骤，创建另一个新的 SLA。在场景中，拥有多个 SLA 是非常常见的。当所有 SLA 都设置完毕后，关闭 SLA 向导，这时控制器中 SLA 区域将变为图 12-19 所示。

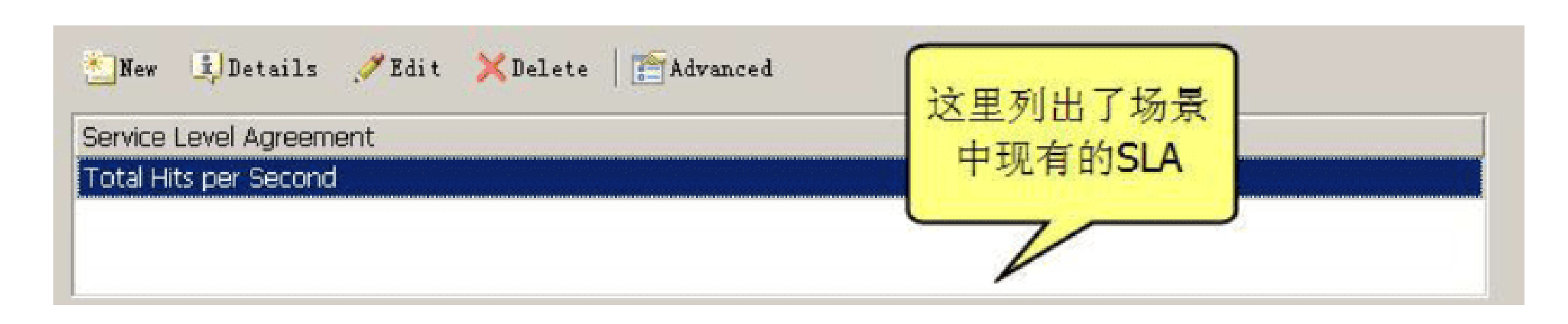


图 12-19　添加完毕后控制器将显示现有 SLA 列表

12.2.4　利用高级按钮设置时间间隔

在图 12-11 和图 12-19 中，都可以发现一个“高级设置”（Advanced）按钮，单击该按钮后将出现如图 12-20 所示的窗体。通过窗体上某些选项的设置，就能够回答本章前面 12.2.2 节中提出的问题，为时间决定的 SLA 设置间隔。

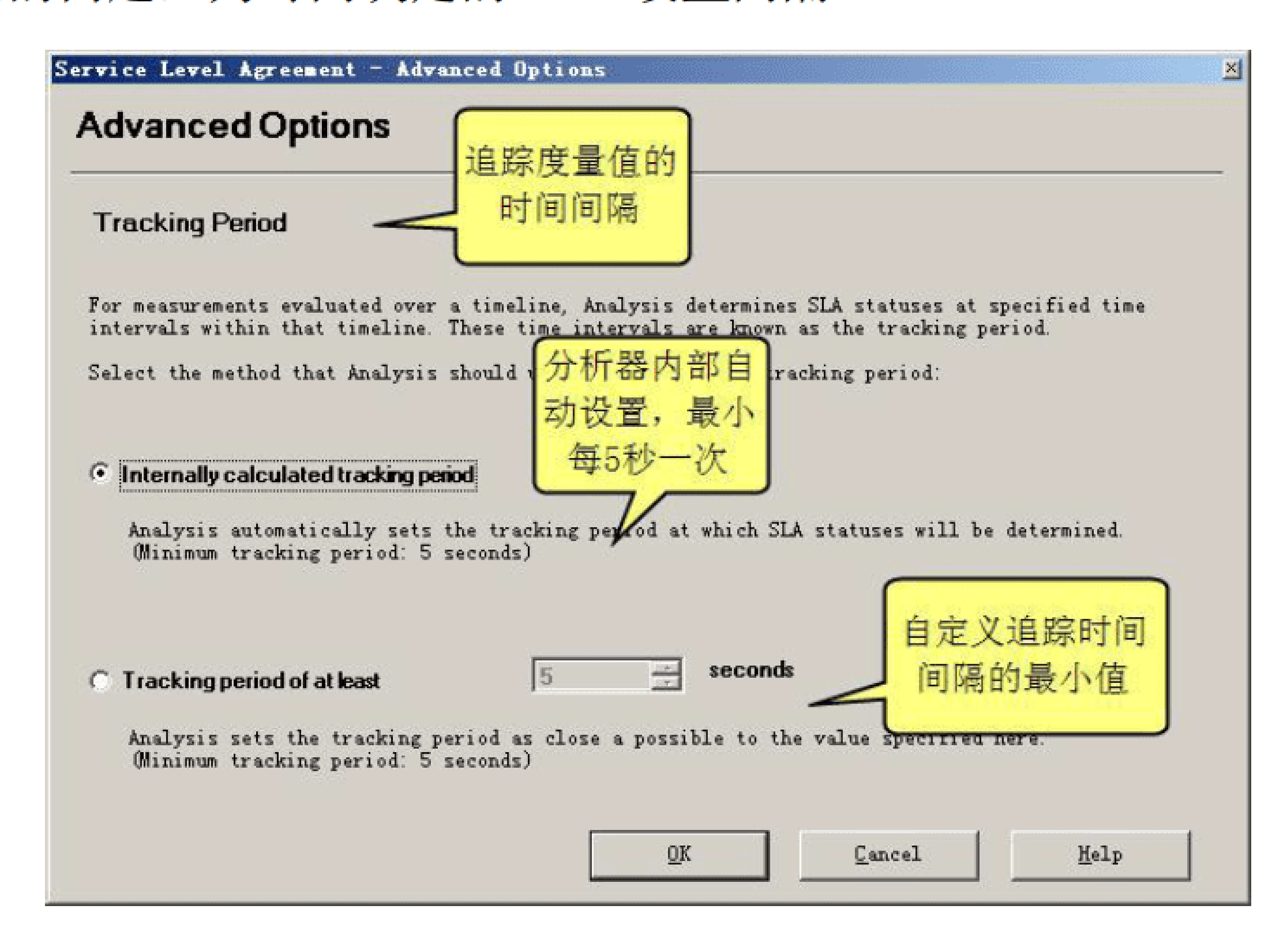


图 12-20　设置追踪时间间隔

【SLA 状态记录间隔与精度】

可以发现，高级选项用于设置图 12-12 中第一个单选框——SLA 状态被时间决定的追踪时间，最小值都是 5 秒，即每隔最短 5 秒会记录一次 SLA 状态（通过、不通过或者无数据）。对于运行决定的 SLA，高级选项是没有作用的。另外，在图 12-20 中有分析器内部计算和测试工程师人工指定两种方式，其中后者设置的时间间隔在运行时只能尽量接近。

在了解 SLA 之后，再回过头来阅读场景执行完毕后分析器显示的测试分析概要，就不会产生疑问了。12.3 节将讲述测试分析概要的解读。

12.3　解读测试分析概要

在性能测试结束之后，默认情况下，LoadRunner 将会自动生成“测试结果概要”（Analysis Summary）页面并自动打开分析器，这在前文中已经介绍过了。对于测试结果，我们首先需要了解的就是总体情况，因此，首先关注分析器中的测试分析概要很有必要。

12.3.1　测试分析概要界面

测试分析概要（Analysis Summary）界面包含了如下几部分信息：

- ❑ 场景名称、运行时间。
- ❑ 测试执行的统计，比如最大运行虚拟用户、最大吞吐量等。

如图 12-21 是一次性能测试结束后分析概要的示意图。

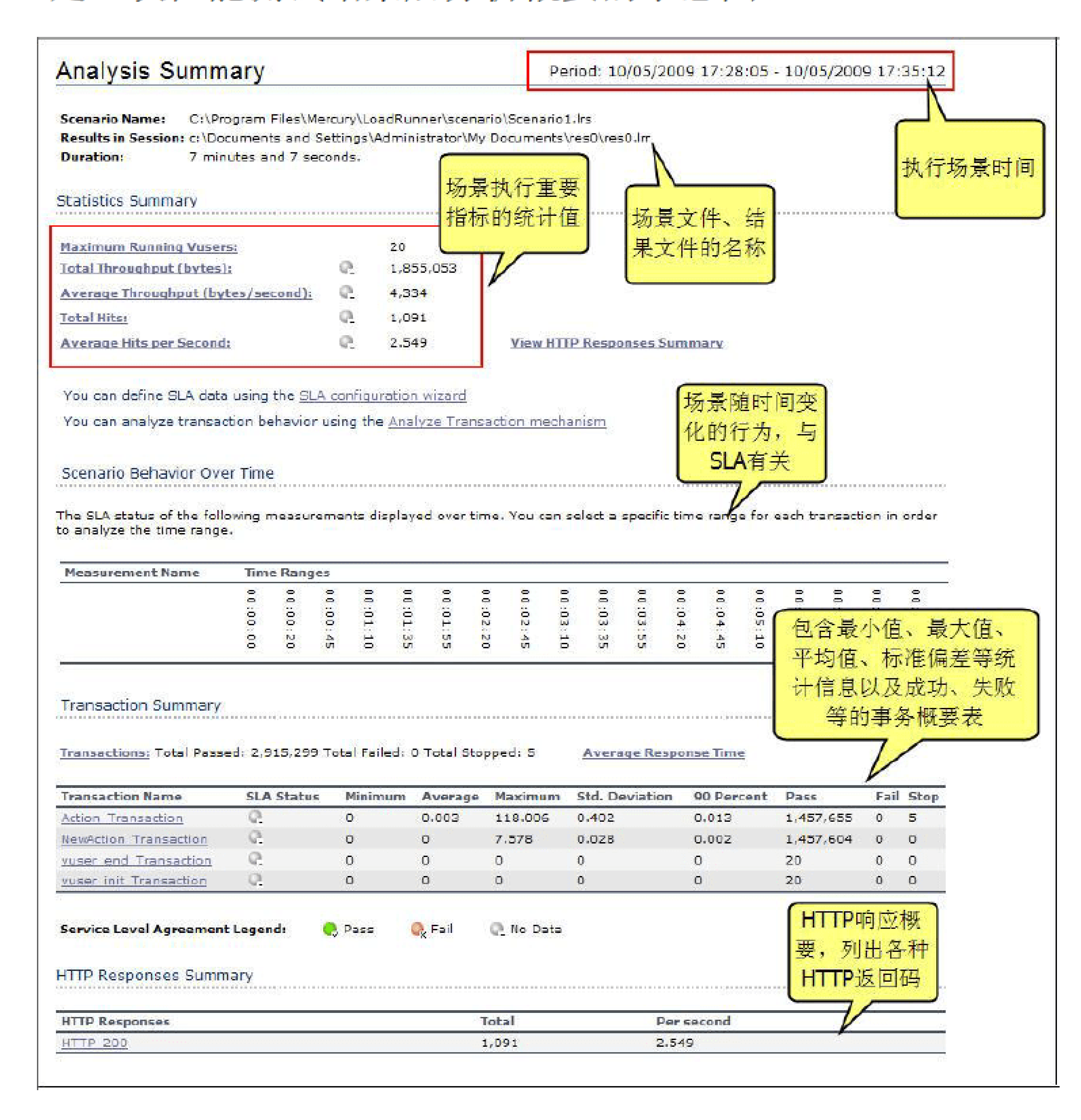


图 12-21　分析器中的测试分析概要

12.3.2　统计概要

该报告的第一部分统计概要（Statistics Summary）描述了场景执行重要指标的统计值，默认是如下几种：

- ❑ 运行中的虚拟用户最大值，在图 12-21 中是 20 个，这是由于沿用了第 11 章中的设置（两个虚拟用户组，每组各 10 个虚拟用户）。
- ❑ 总吞吐量（单位：字节数）。
- ❑ 总点击量，以次数为单位。
- ❑ 每秒点击次数。在其后还有查看 HTTP 响应概要的链接，单击后将直接跳转到报告的最后一部分，将在下文简单介绍。

注意，这几个默认统计值都是包含链接的，因此，都可以单击后获得更多信息。在这

里，我们以总吞吐量为例。单击链接后，分析器增加了一个新的选项卡，名为 Throughput（吞吐量），内容则是吞吐量变化图和一些统计信息，如图 12-22 所示。

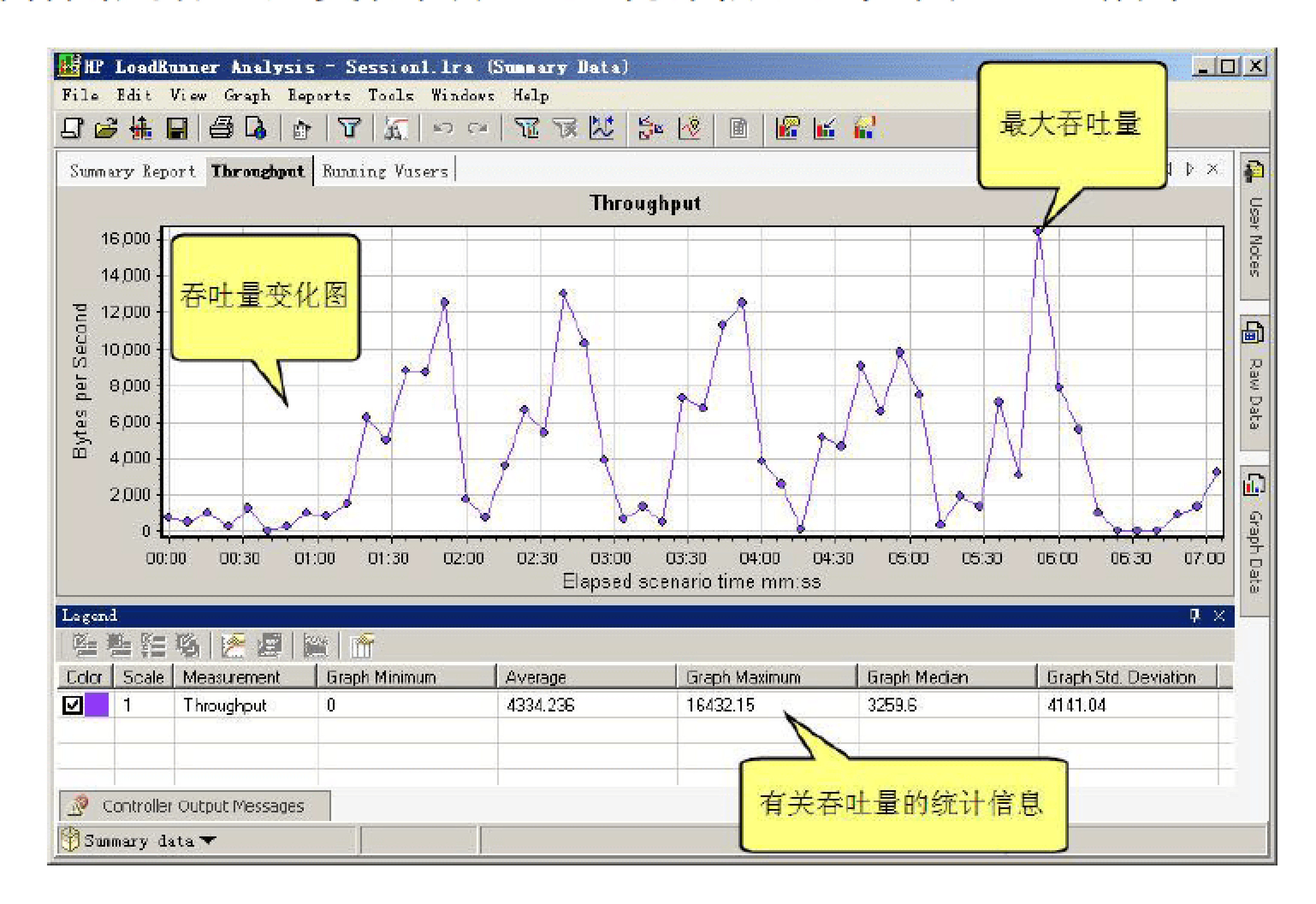


图 12-22　单击统计概要中总吞吐量链接后出现吞吐量变化图

有关各项图表的分析，将在 12.3.3 节介绍。对于统计信息，分析器列出了最小值、最大值、平均值、中位数、标准偏差等多项。关于这些数值的计算方法，在本书的第 13 章均有介绍，这里可以直接采用 LoadRunner 给出的计算结果即可。

紧随统计概要之后，报告的第二部分是场景执行过程信息，它与 12.2 节所介绍的服务质量协议（Service Level Agreement，简称 SLA）密切相关。

12.3.3　场景执行过程信息表

统计概要报告的第二部分是场景执行过程信息。在控制器中是否设置了 SLA，将决定此处显示的信息会有所不同。如图 12-23 和图 12-24 分别显示了同一个场景中没有 SLA 和包含 SLA 的结果，请读者注意它们的区别。

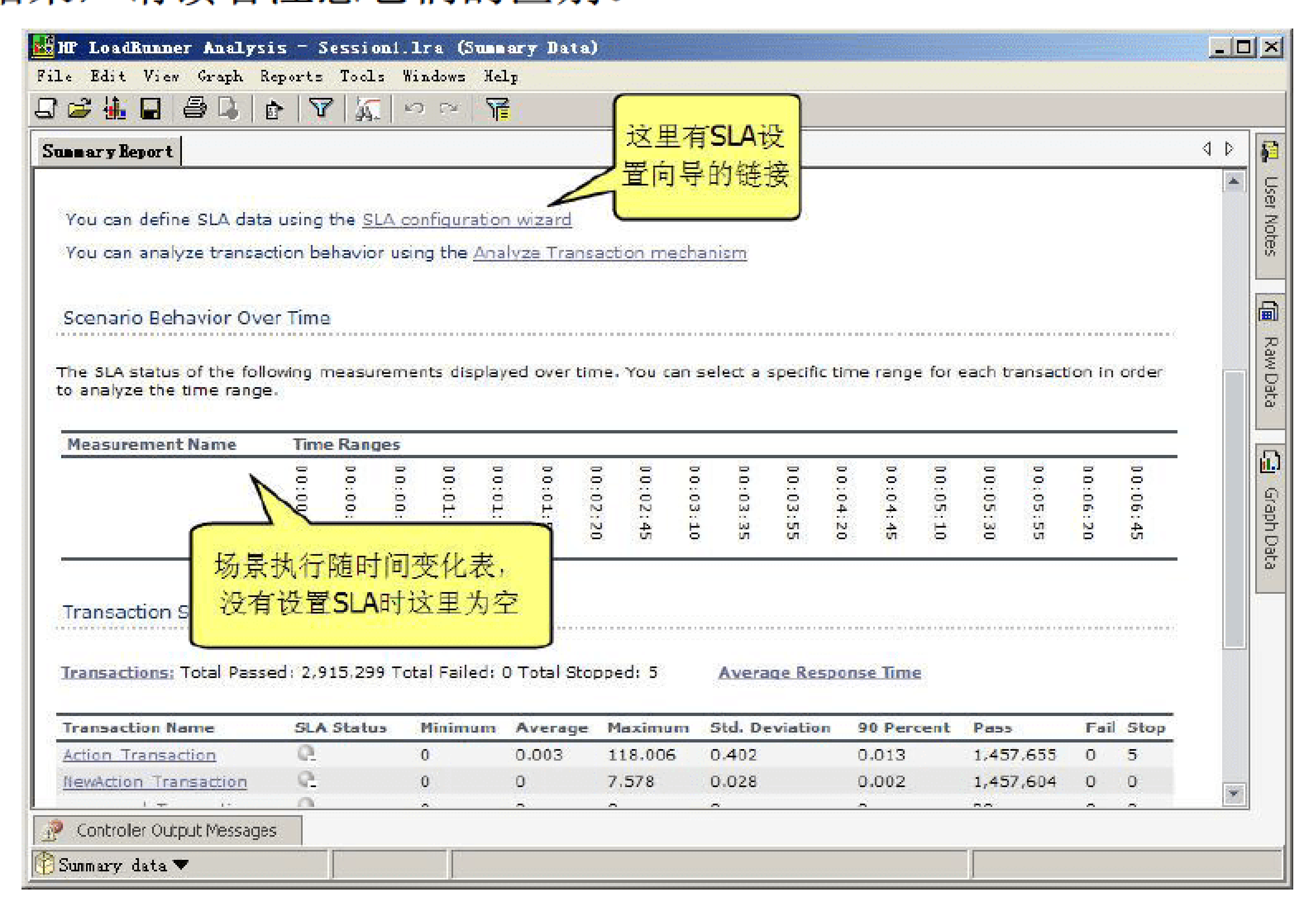


图 12-23　场景中未设置 SLA 后的场景执行过程信息表

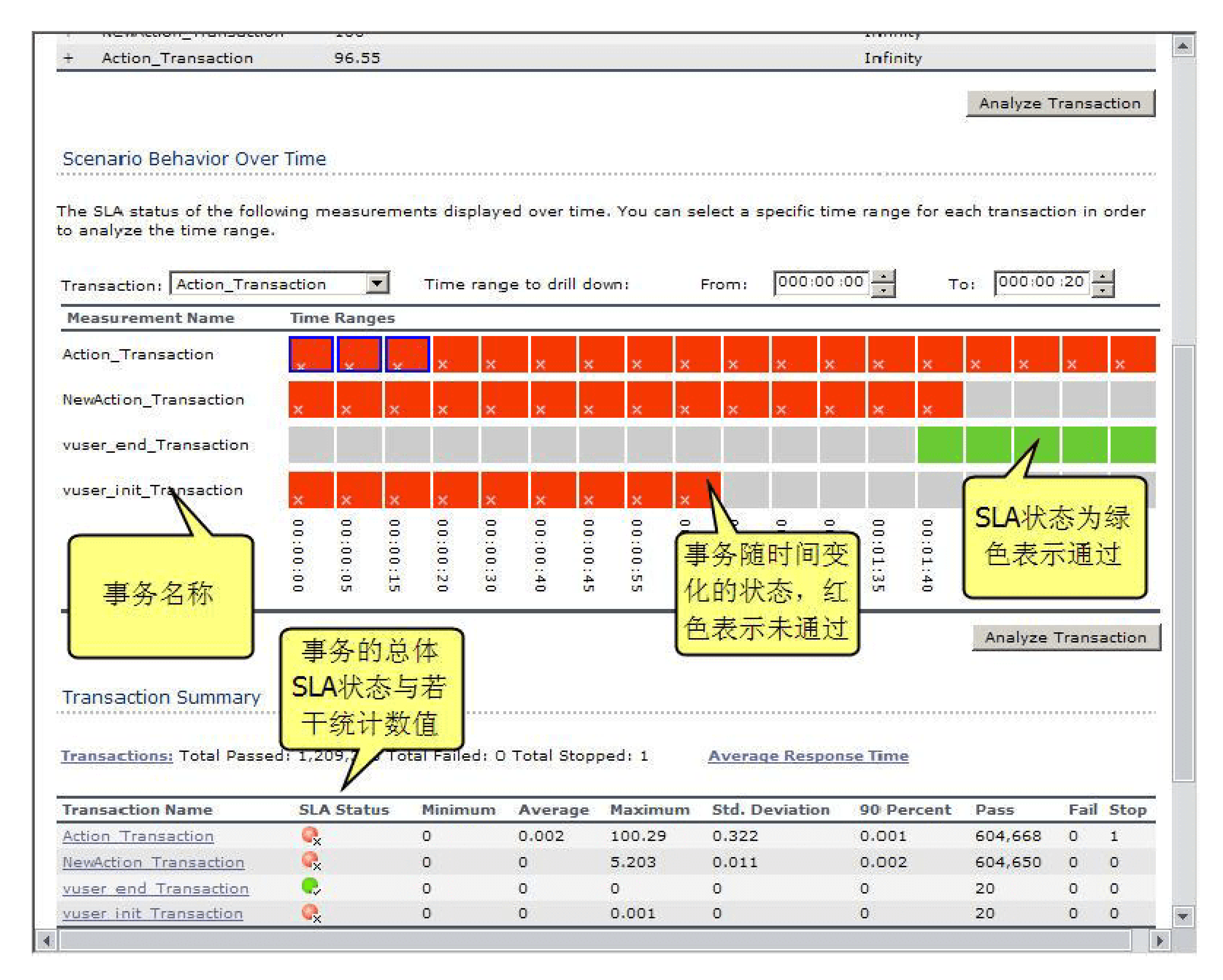


图 12-24　设置了 SLA 的场景运行后统计概要界面

【设置 SLA 与否导致统计概要界面有区别】

如果在场景中未设置 SLA，则统计概要报告中会列出 SLA 设置向导的链接，供修改场景之用（当然，场景并不是必须包含 SLA）；与此同时，场景执行过程信息表的度量值名称（Measurement Name）栏将为空。实际上，若有内容，这里列出的将是应用了 SLA 的事务名称，请参看图 12-24 中的相同位置。

12.3.4　对事务进行 SLA 相关分析

在图 12-24 中，读者可以很方便地查看随运行时间，场景中应用了 SLA 的各个事务运行情况。红色表明 SLA 没有通过，而绿色则表明达到了 SLA 的要求。右击每一个表明 SLA 状态的方框（即图 12-25 中每一个鼠标悬停时会变为手状的方格），在弹出的快捷菜单中还可以查看这一时段内事务与 SLA 的具体情况，比如 Analyze SLA（分析 SLA）选项，如图 12-25 所示。

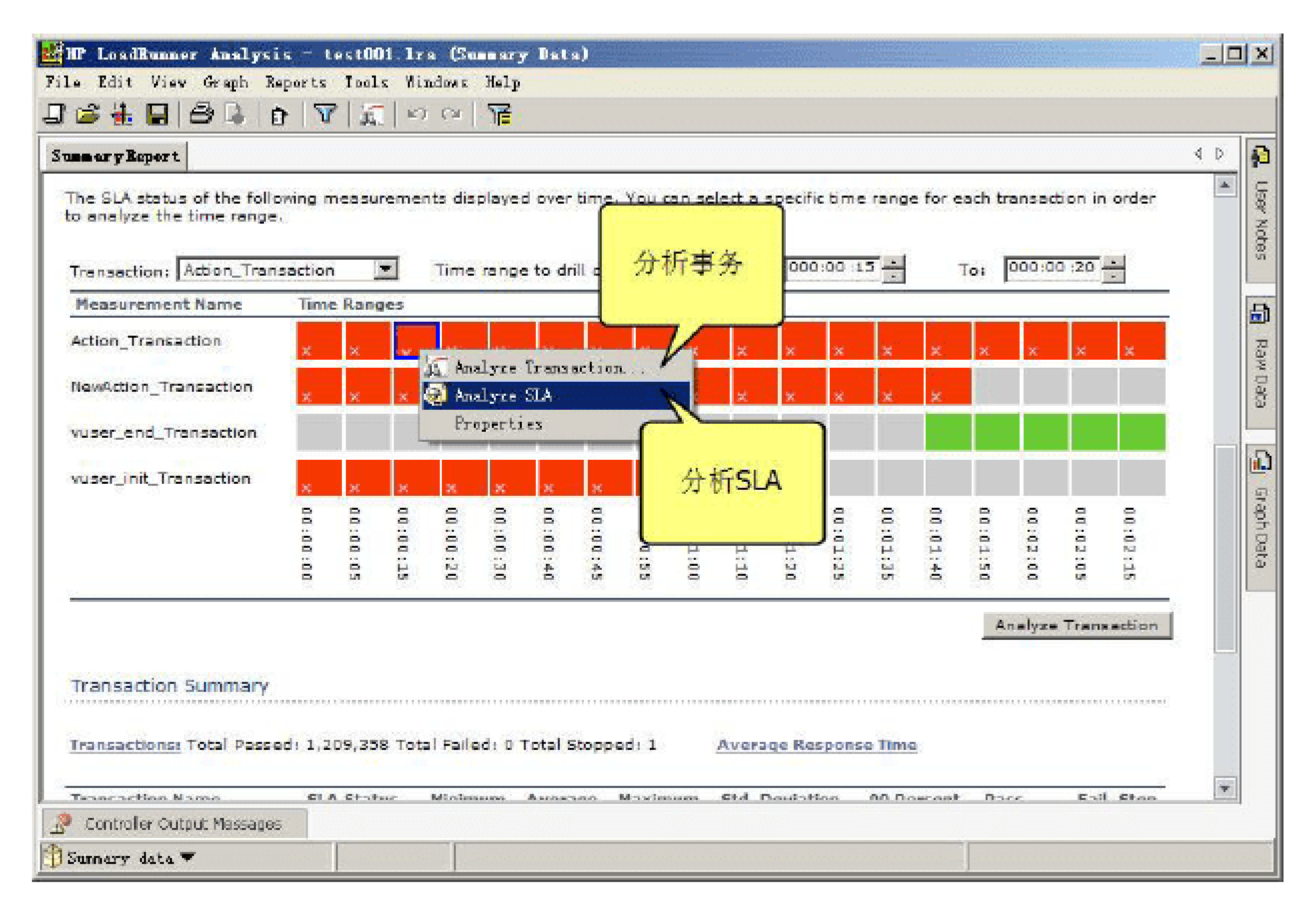


图 12-25　右击每个 SLA 状态后的弹出菜单

【对事务进行分析】

选择图 12-25 中右键弹出菜单下的 Analyze Transaction（分析事务）选项，将弹出相应窗体，如图 12-26 所示。该图的左边部分默认会列出各个事务中未通过 SLA 的时间段。当测试工程师选择某一个事务或者其中的某一段时间后，右边的事务图将跟随更新，反映场景执行时，该事务随时间的变化。

在右边的事务图中，包含 3 种曲线，分别为：

- 与事务相关的对照曲线，可以是如下 3 种，其变化趋势可以用于辅助分析为何未通过 SLA 时的对照。
  - 运行的虚拟用户（Running Vusers）；
  - 吞吐量（Throughput）；
  - 每秒点击量（Hits per second）。
- 实际值曲线（图例中标识为事务名称的那条颜色曲线）。
- 期望 SLA 值的曲线（Expected SLA）。

关于对照曲线中的 3 个指标，若能进行正确选择，将对发现随负荷增长而未通过 SLA 的原因很有益处。具体采用何种图表进行比对，以及从图表的变化趋势中发现性能问题将是 12.3.5 节的内容。

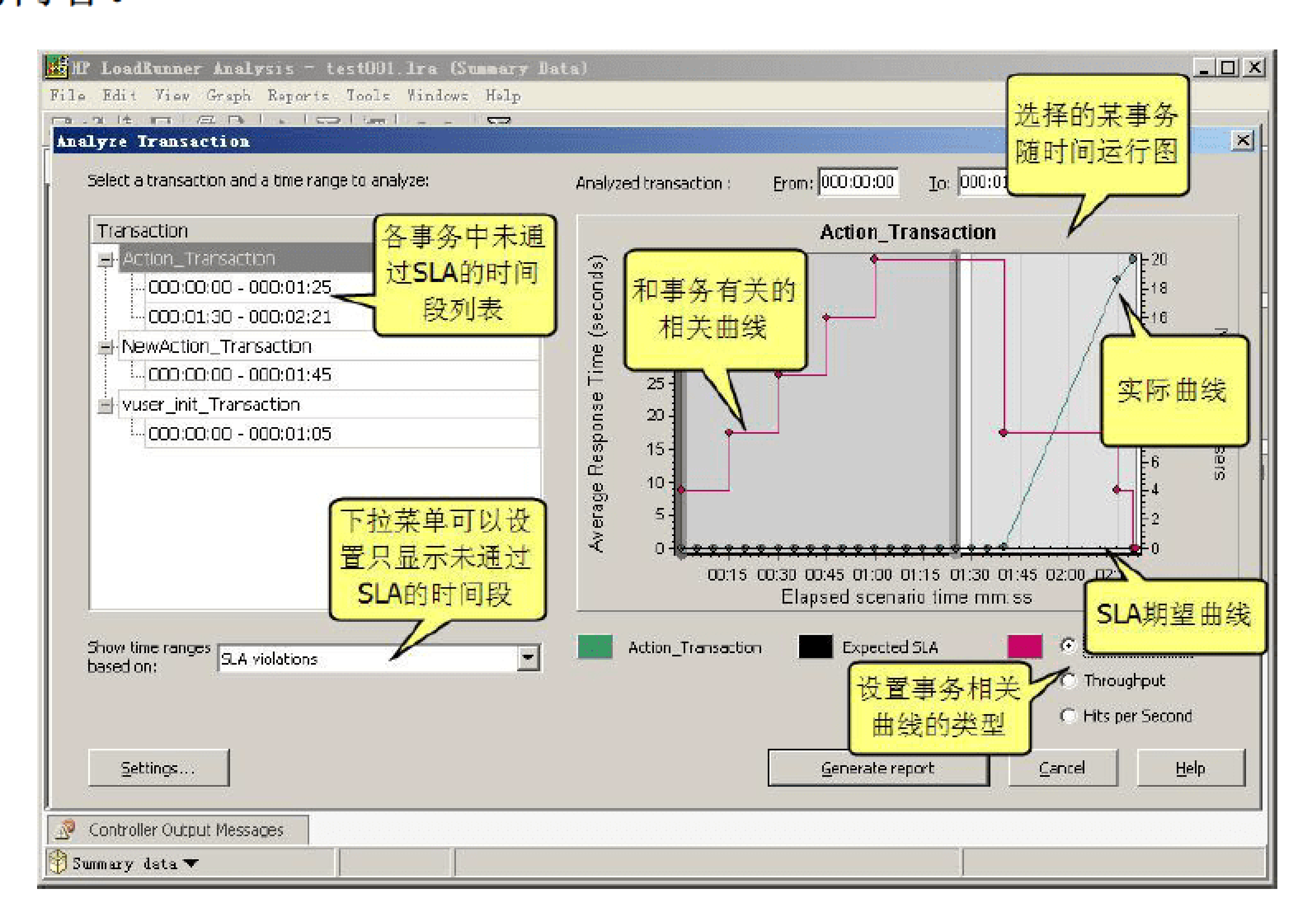


图 12-26　分析事务窗体

12.3.5　分析 SLA

在图 12-25 的某方格中右击，在弹出的快捷菜单中选择 Analyze SLA 选项，将弹出 SLA 报告页面，如图 12-27 所示。SLA 报告总体分为两个部分：

（1）各个 SLA 设置值与实际值以及 SLA 运行时设置信息显示。在产生图 12-23 所示结果的场景中我们总共添加了两个 SLA，即在 12.2 节中举例设置的总点击数和事务响应时间。因此，在 SLA 报告中也列出了 2 组期望值与实际值。由于均为达到期望值，所以在 SLA 报告的开头，状态显示为未通过（Failed）。

（2）SLA 运行时的设置信息，比如查询度量值的间隔（默认是 5 秒），各事务的达标虚拟用户数等，都会清楚地列在 SLA 报告之中。



图 12-27　SLA 报告

至此，测试分析概要中场景执行过程信息表就介绍到这里。我们将继续熟悉分析概要中的下个部分——事务概要。

12.3.6　事务概要

统计概要页面的第 3 个部分是事务概要（Transaction Summary）部分，如图 12-28 所示。这部分信息是前一部分——场景执行过程信息表的另一种总结，忽略了时间因素，只按照场景中包含的事务列表来显示各种统计信息。每一个事务同时也是一个链接，可以单击打开，如图 12-29 所示。



图 12-28　事务概要表

【不同操作可产生相同的图表】

可以发现，图 12-29 与图 12-26 中的曲线图是完全一致的，只是在不同的界面中，呈现的方式不同。由于图 12-29 增加了选择功能，可以屏蔽掉不想关注的某个或某几个事务，在曲线图中不再列出。在实际操作中，只要不选择事务之前的复选框即可。

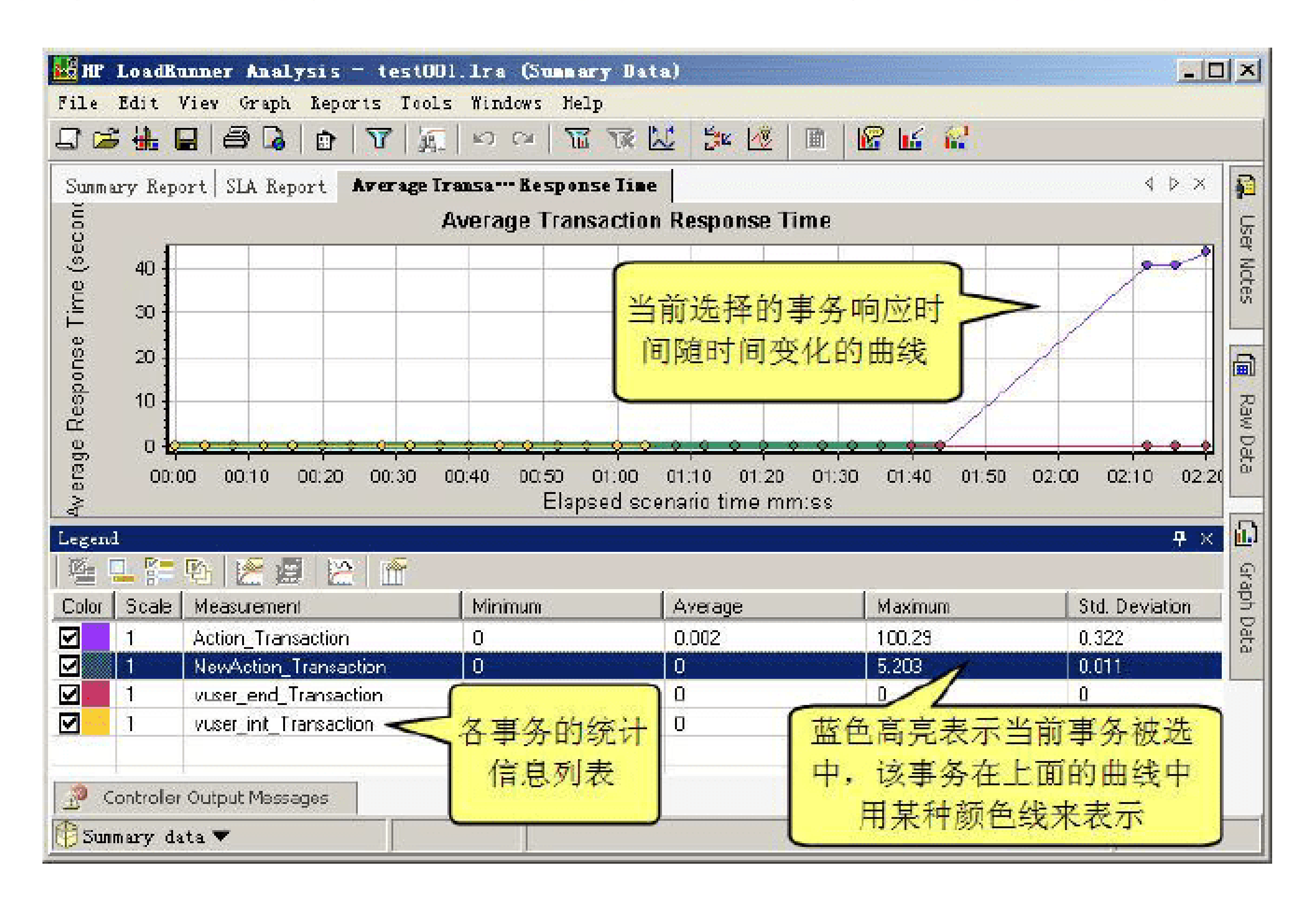


图 12-29　事务响应时间曲线图

为了使图 12-29 中的显示曲线适合自己的习惯，可以对其进行设置。比如要修改各事务曲线的颜色，可以单击曲线图与事务列表之间 Legend（图例）工具栏的 Configure Measurements（配置度量）图标按钮，其后弹出设置窗口如图 12-30 所示。另外，单击图例工具栏中的 Configure Columns（配置列）图标按钮，还可以显示与隐藏事务统计信息表中显示的各个数据列，如图 12-31 所示。

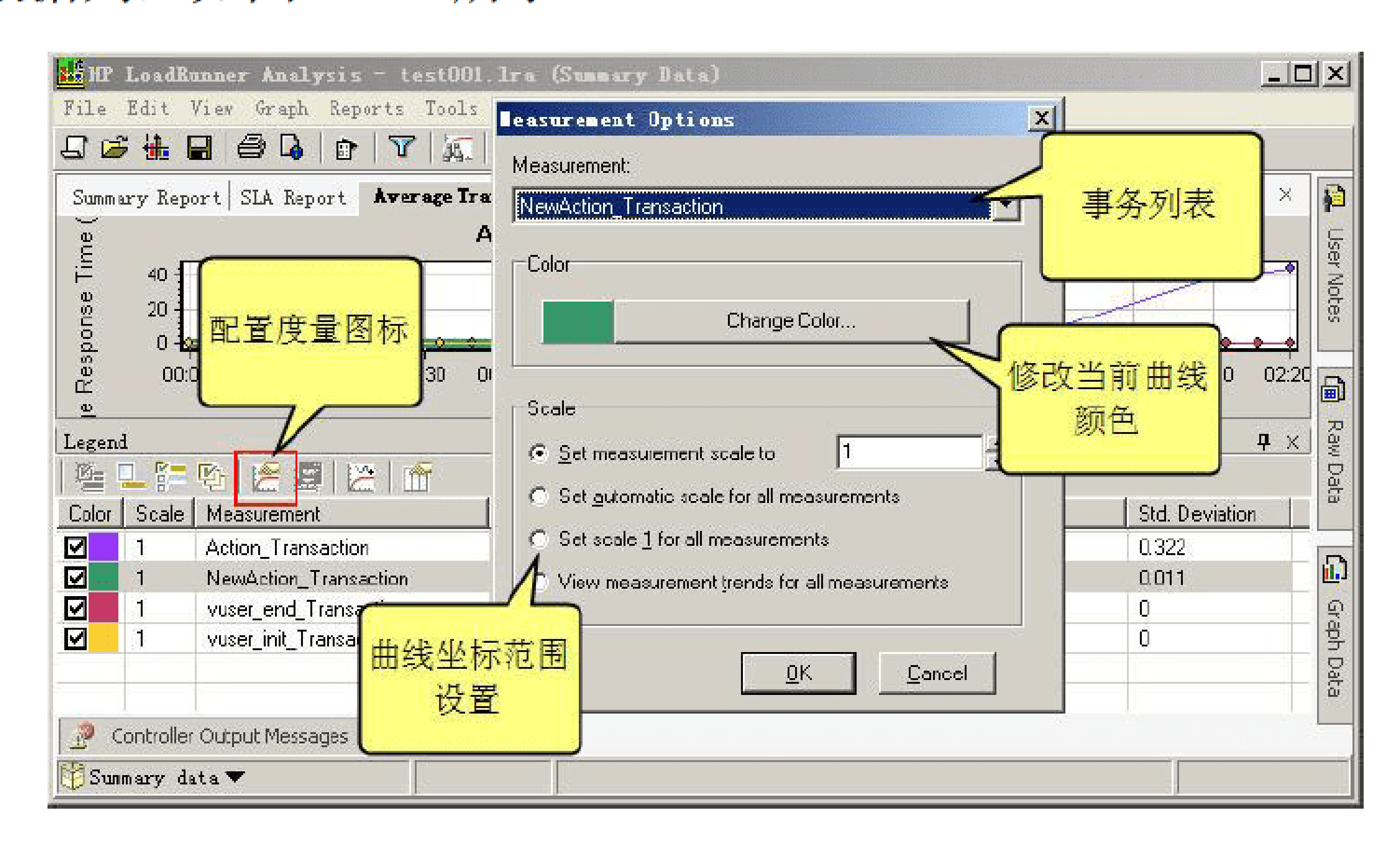


图 12-30　修改曲线的度量设置

【增强图形可读性的设置】

图例工具栏中还有一些方便使用者读曲线图的图标按钮，比如动画显示所选择曲线、隐藏无关事务曲线等，读者可以在实际工作中根据需要选用。因为比较简单，在此就不介绍了。

截至目前，本章已经介绍了很多图表，可能令读者感到混淆。实际上，由于所有的图表都是基于相同场景运行的同一份结果数据，为了不同的分析方式、方法的便利，LoadRunner 将这些数据在不同的界面中进行展示，以反映一个或几个方面的特点。

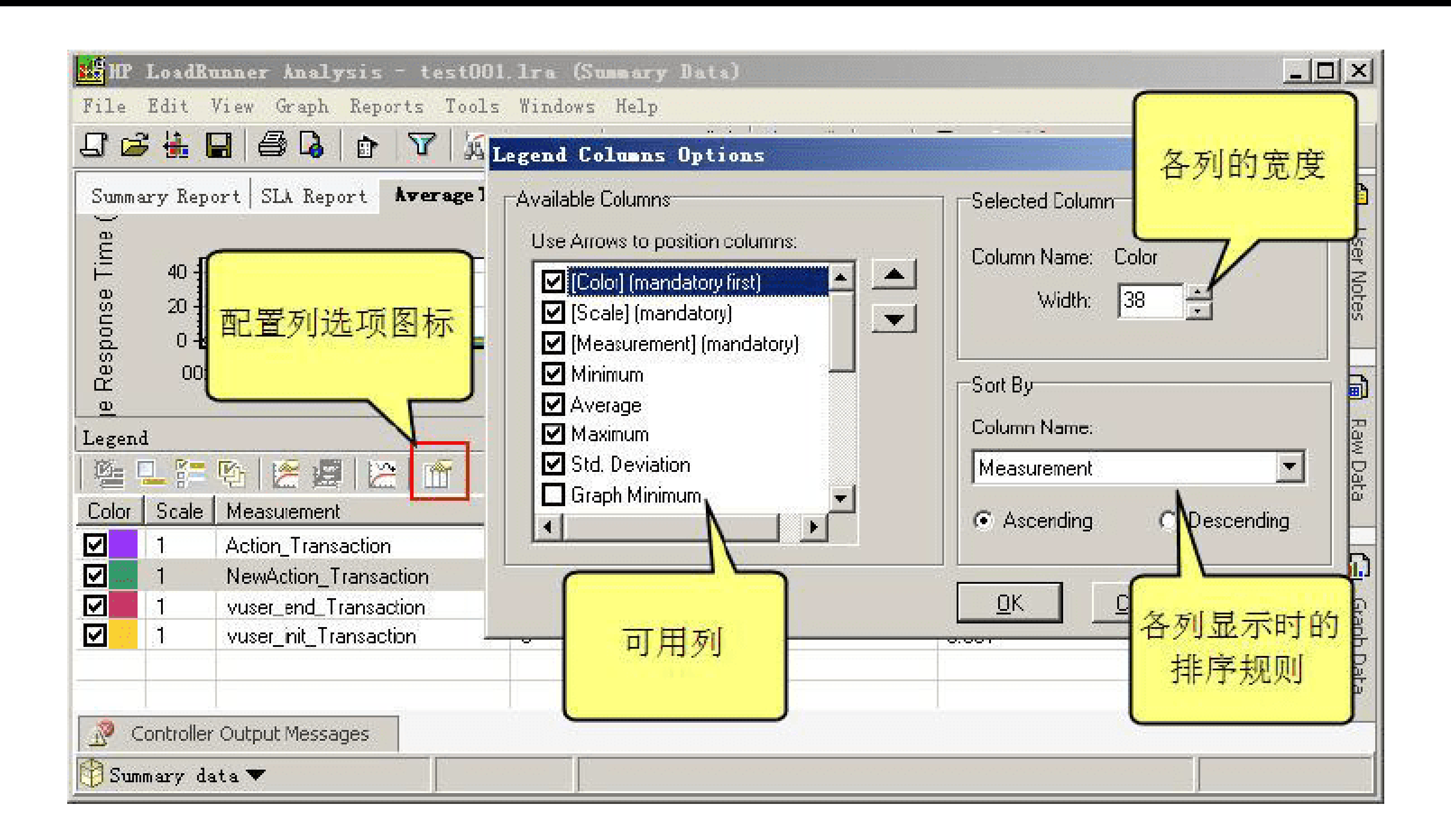


图 12-31　配置事务列表的列选项

12.3.7　HTTP 响应概要

统计概要报告最后的部分是 HTTP 响应概要，其中会列出各种 HTTP 响应代码的分布。由于本节所举例执行的场景各步骤均成功，所以在图 12-32 中仅仅列出了 HTTP 响应代码为 200（表示成功）的数据。

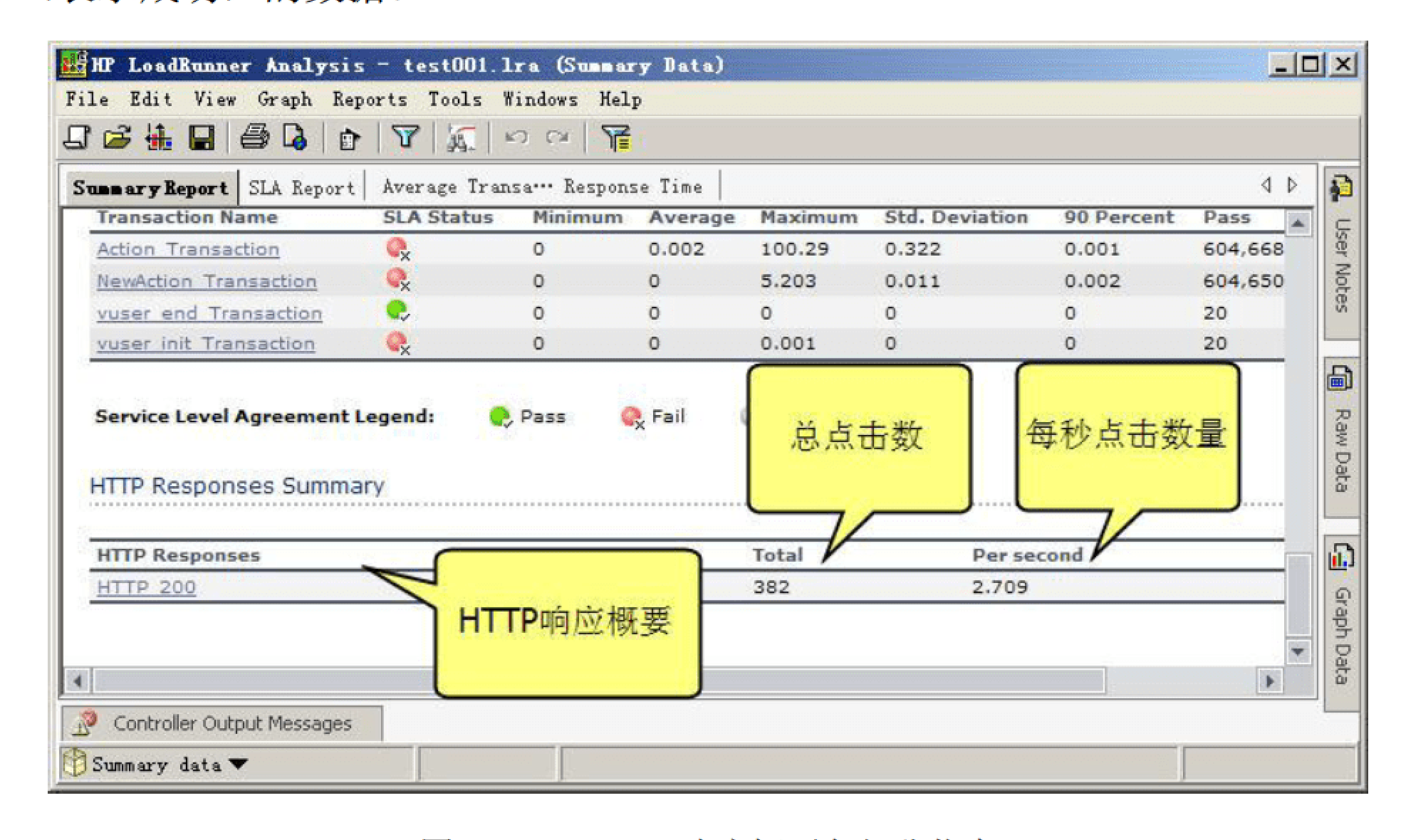


图 12-32　HTTP 响应概要各部分信息

【HTTP 响应码】

响应码由 3 位十进制数字组成，它们出现在由 HTTP 服务器发送的响应的第一行，用来表示状态，便于浏览器、服务器以及其他相关程序通信。具体的 HTTP 响应码及其含义请见表 12-2 所示。

与前面介绍过的图表类似，单击每一个 HTTP 响应代码，都可以进一步查看该次执行中该响应码的具体信息，如图 12-33 所示。

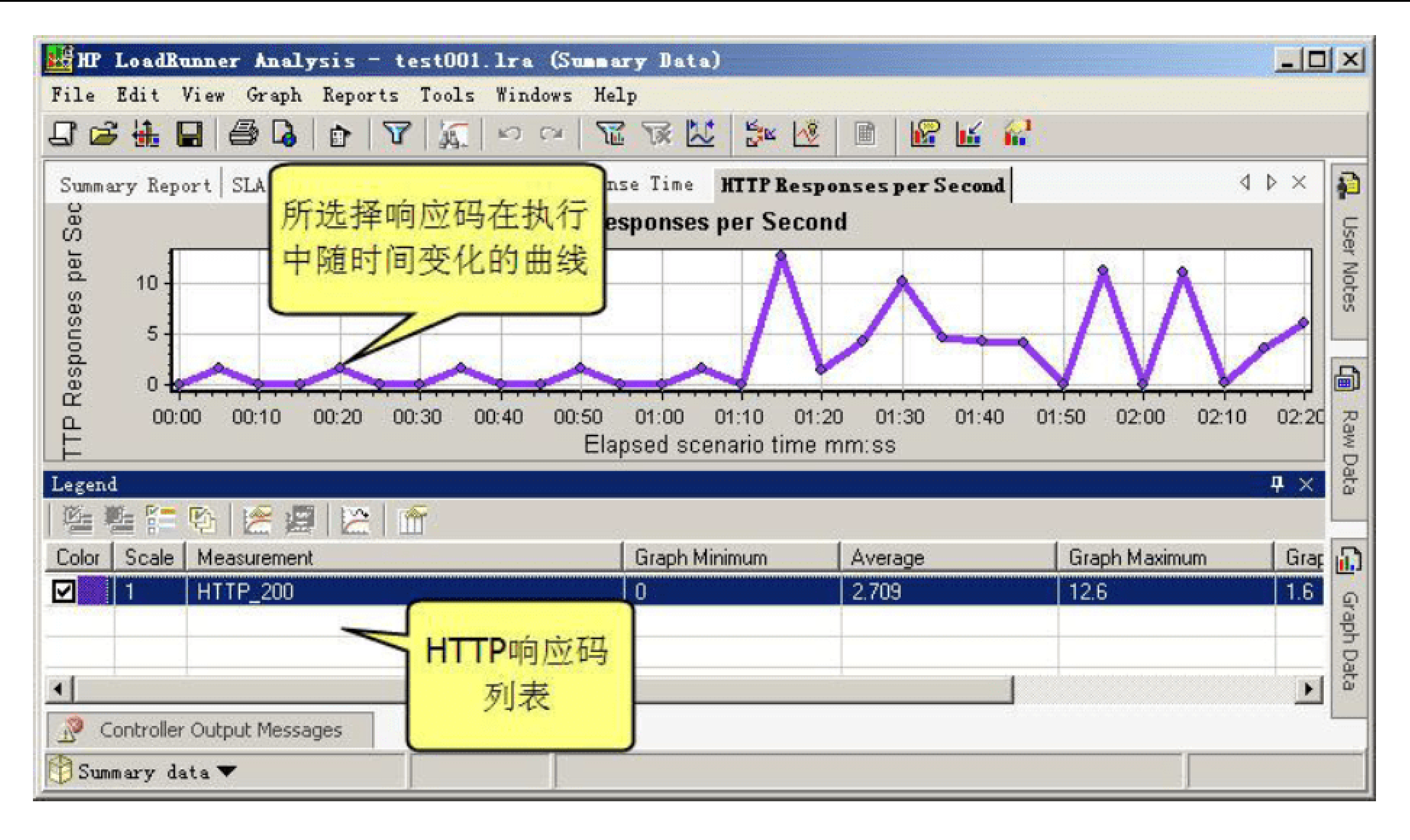


图 12-33　每秒 HTTP 响应变化图

可以注意到，在响应码列表与曲线之间，同样有 Legend（图例）工具栏，其中的各个图标与前面介绍过的一致，可以做类似的修改和设置。

在实际情况中，只有 HTTP 200 的情况是非常少见的，随着虚拟用户的增加，负荷的增大，各种响应返回码都可能出现。如表 12-2 列出了常见的 HTTP 返回码以及代表的含义。HTTP 返回码共分为 5 类，分别用数字 1、2、3、4、5 开头。

表 12-2　常见HTTP返回码

分　　类	返　回　码	说　　明
1xx：信息代码	100	继续
	101	分组交换协议
2xx：成功代码	200	OK
	201	被创建
	202	被接受
	203	非授权信息
	204	无内容
	205	重置内容
	206	部分内容
3xx：重定向代码	300	多选项
	301	永久地移除
	302	找到
	303	参见其他
	304	未改动
	305	使用代理
	306	（未使用）
	307	暂时重定向
4xx：客户端错误代码	400	错误请求
	401	未授权
	402	要求付费

续表

分　类	返　回　码	说　明
4xx：客户端错误代码	403	禁止
	404	未找到
	405	不允许的方法
	406	不可接受
	407	要求代理授权
	408	请求超时
	409	冲突
	410	过期的
	411	需要长度信息
	412	前提不成立
	413	请求实例太大
	414	请求 URI 太长
	415	不支持的媒体类型
	416	无法满足的请求范围
	417	预期失败
5xx：服务器错误代码	500	内部服务器错误
	501	未实现
	502	错误网关
	503	服务不可用
	504	网关超时
	505	HTTP 版本未被支持

以上这些 HTTP 代码，特别是 HTTP 200、HTTP 404、HTTP 403、HTTP 500 等都是在实际测试中经常遇到的响应码。

截止目前为止，本节已经把测试分析概要的几个部分介绍完毕。场景执行所生成的图表远远不止以上反应基本信息这些，在第 13 章中，我们将重点学习更多的图表以及通过对各图表的合并、关联等技巧来分析和发现性能问题。

12.4　本 章 小 结

本章可以分为 2 大部分：LoadRunner 场景的执行和运行结束后自动生成的统计概要页面。前者是在控制器（Controller）中进行设置和启动、停止的；后者则在分析器（Analysis）中打开。读者完全可以修改默认设置，使得运行后不自动生成分析概要。

在场景的执行中，特别需要注意的是 LoadRunner9 的一项新功能：服务质量协议（SLA）。SLA 实际上是某项指标的期望值，当没有达到期望值（具体到各个指标的不同，达标标准可能是超过或者不足）的时候，SLA 状态将标记为失败。分析概要能很好地显示 SLA 状态。

分析概要中还包括若干图表，可以为了解运行情况和总体性能提供基本数据。但是，只依靠分析概要中的图表是不够的，还需要对图表进行多种分析。

第 13 章笔者将介绍如何添加更多的图表、如何利用合并、关联等技巧其进行分析并从中发现性能问题。

第 13 章　分 析 结 果

截止目前为止，我们已经接触到 3 个 LoadRunner 重要组件，分别如下。

- 虚拟用户生成器（VuGen）：用于录制脚本等功能。
- 控制器（Controller）：用于场景和虚拟用户组的生成和配置。
- 分析器（Analysis）：用于分析场景运行结果。

以上 3 个组件既可以在彼此的界面内部打开，也可以独立运行，如图 13-1 所示。

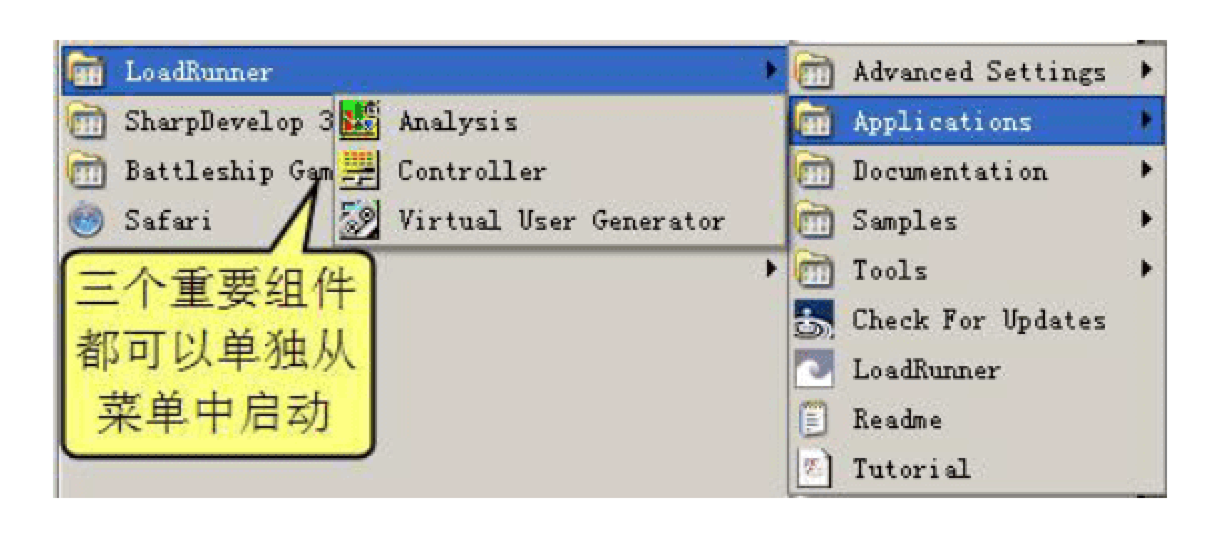


图 13-1　可以直接在菜单中启动各组件

LoadRunner 的分析器将每一次分析过程称为一次分析会话（Session），大多数情况下并不是很短的时间可以完成的。因此测试工程师可以将其保存成后缀名为 lra（LoadRunner Result Analysis）的文件，从而保存思路，更可以储存多次分析结果，方便项目管理。

本章将在前面介绍的分析器的知识基础上，重点讲述如何利用图表发现性能问题。

13.1　分析器简介

在桌面单击“开始”|“程序”|LoadRunner|“应用程序”|“分析器”，即可打开分析器界面。我们可以选择上一次保存的 lra 文件，继续对上一次的测试结果进行分析。

13.1.1　分析器界面的几大部分

分析器界面如图 13-2 所示，共分为如下几个部分。

- 图表显示区域：通过多个标签页来显示多个图表，可以自定义添加或者删除图表。这里的删除并不会删除数据，只是不在界面上显示。
- 原始数据（Raw Data）显示：包含一个链接用以显示原始数据。
- 图表数据表（Graph Data）：列出当前显示图表所包含的 X、Y 轴数据。
- 用户说明（User Notes）：测试工程师可以将一些信息、备忘和说明填写到该文本框中，便于日后在工作中查看。

- 图例（Legend）与曲线选择：列出了当前显示图表中各曲线的统计信息以及一些相关操作（具体可以参看第 12 章图例说明部分）。
- 控制器输出（Controller Output）信息：包含一个链接单击后可以显示场景运行的相关信息。
- 菜单和工具栏：在对图表进行分析时非常有用，将在 13.2 节中进行详细讲解。

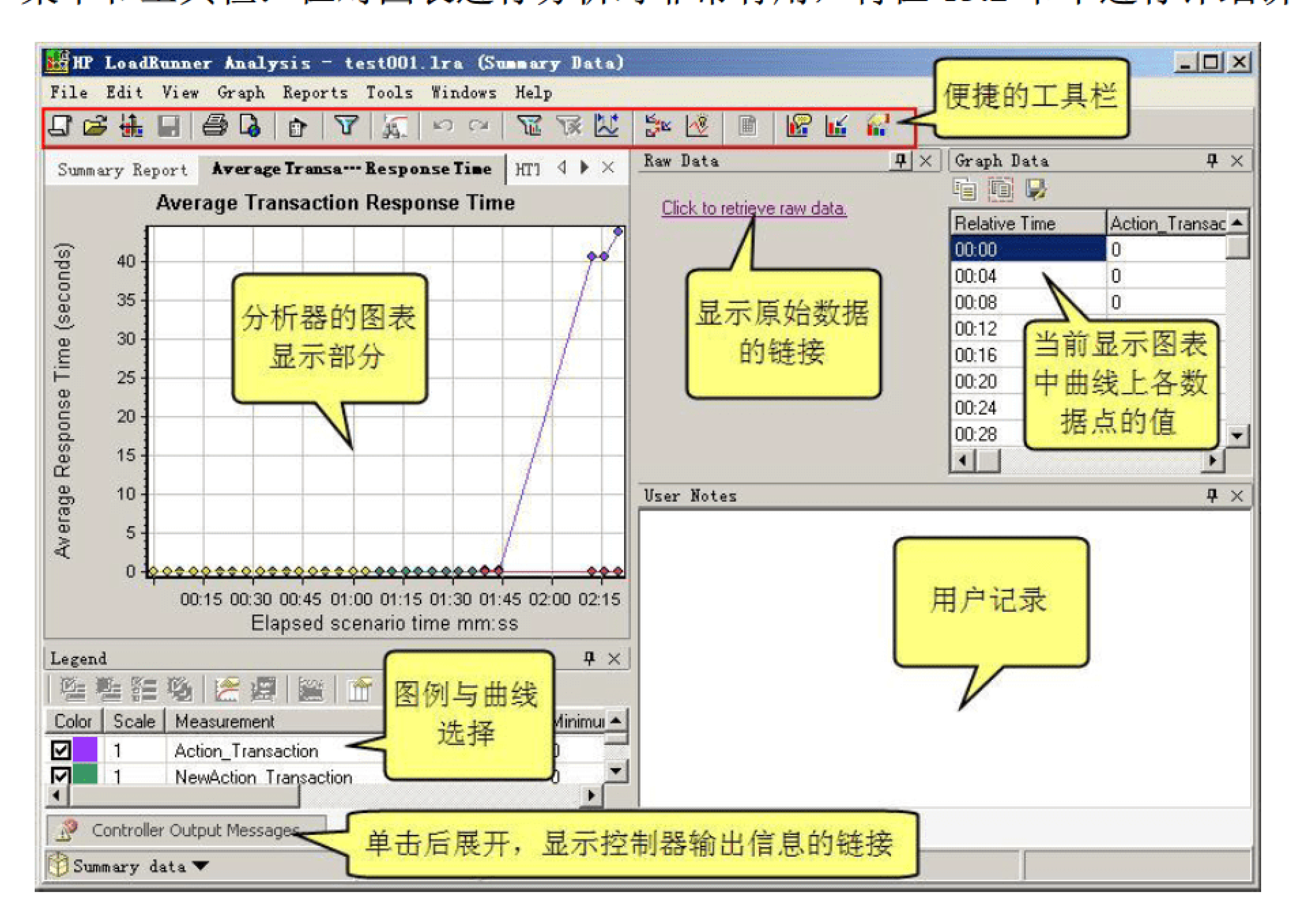


图 13-2　分析器界面的几大部分

分析器最主要的作用是为性能测试工程师分析图表服务的，将在本章的 13.2 节详细讲解。除此之外，分析器还有一些很有意思的菜单项，它们在实际应用中可能不那么引人注意，有时候却能起到重要作用，比如下面的问题：

- 分析器中能方便地修改场景的某些属性吗？
- 分析器中能定义测试结果报告格式吗？
- 可以自行处理分析器中的测试结果数据吗？
- 分析器中显示的测试数据保存在什么地方？
- 分析器如何与 HP 其他的质量控制软件共享数据？

下面将逐一介绍这几个使用技巧。

13.1.2　在分析器中修改场景属性

有时候，场景已经执行完毕，却发现有些因素没有考虑完全，还需要对场景的某些设置进行修改，这时如果重新打开控制器，再找到场景文件是比较麻烦的。为了方便，LoadRunner 分析器中也提供了一些修改场景的菜单项。

选择 File|View Scenario Run Time Settings（文件|查看场景运行时设置）命令，如图 13-3 所示。在弹出的窗体中可以对场景的迭代信息、场景运行计划进行修改，甚至还可以查看脚本内容，这样就把整个测试相对零散的设置串联起来，如图 13-4 所示。

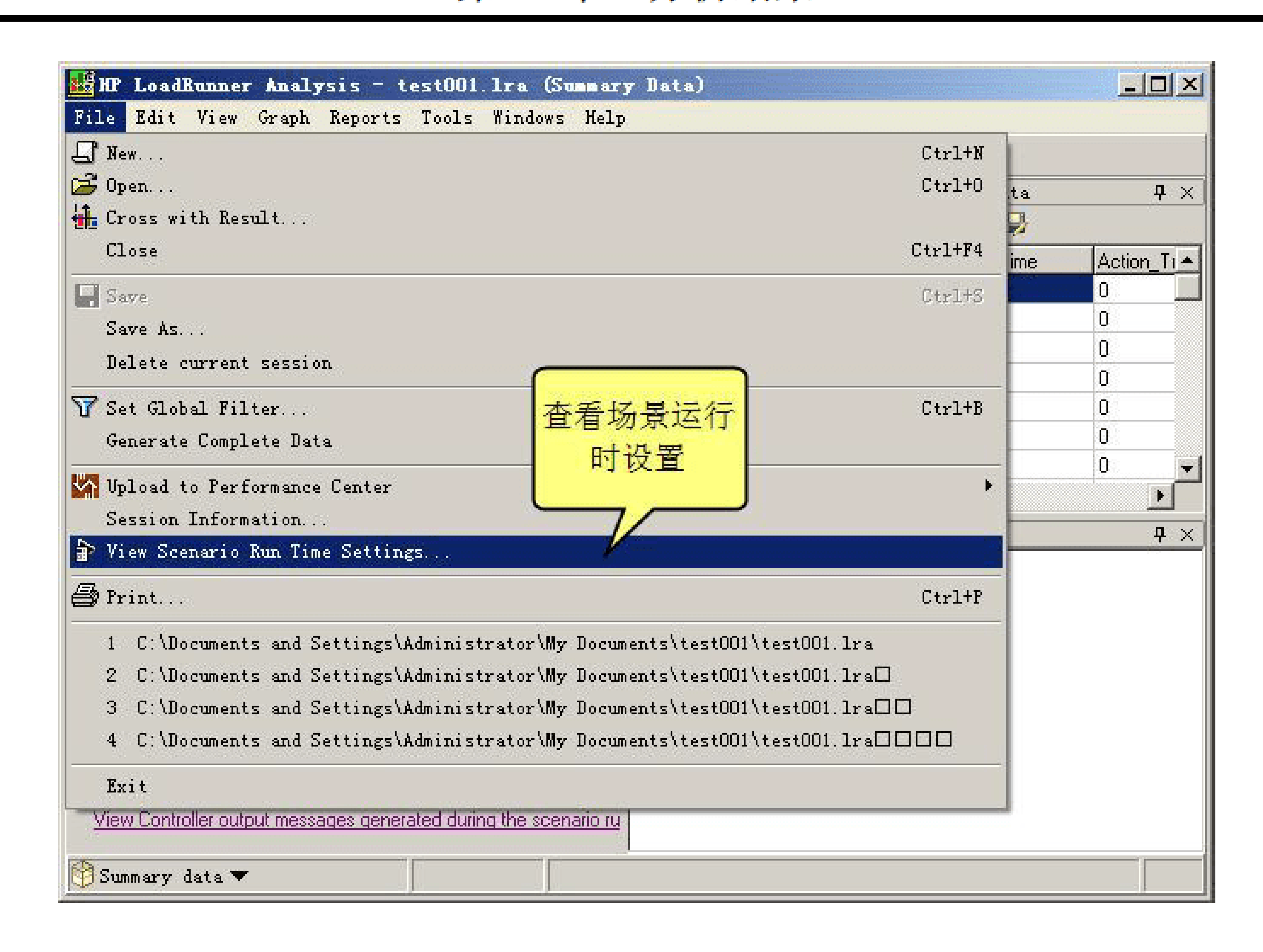


图 13-3　在分析器中查看场景运行时设置

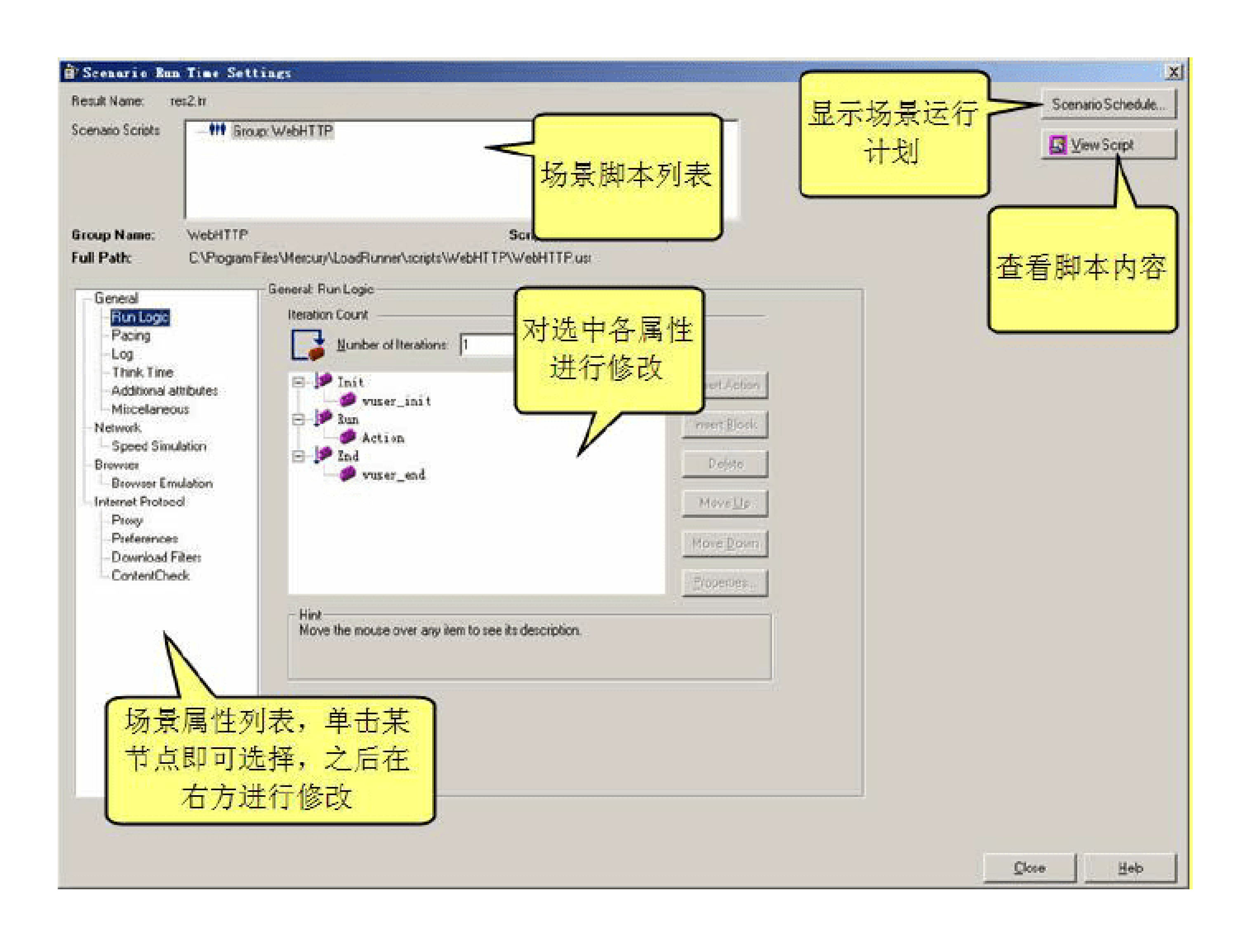


图 13-4　在分析器中打开的场景运行时设置窗口

从图 13-4 中可以看出，其与在控制器中打开的场景运行时设置窗体大同小异。有了这项功能，在分析器中也可以随时修改场景，不必来回切换。

【分析器设置 SLA】

在分析器的工具（Tools）菜单，还有一个菜单项可以对场景的服务质量协议（SLA）进行设置。由于单击菜单项后弹出界面与第 12 章中从控制器打开的完全一样，在此就不单独讲解了。

13.1.3　定义测试报告格式

在测试结束之后，测试报告是一定要发给相关人员的。LoadRunner 通过报告（Report）菜单下的多个子菜单项来支持多种报告的格式，如图 13-5 所示。

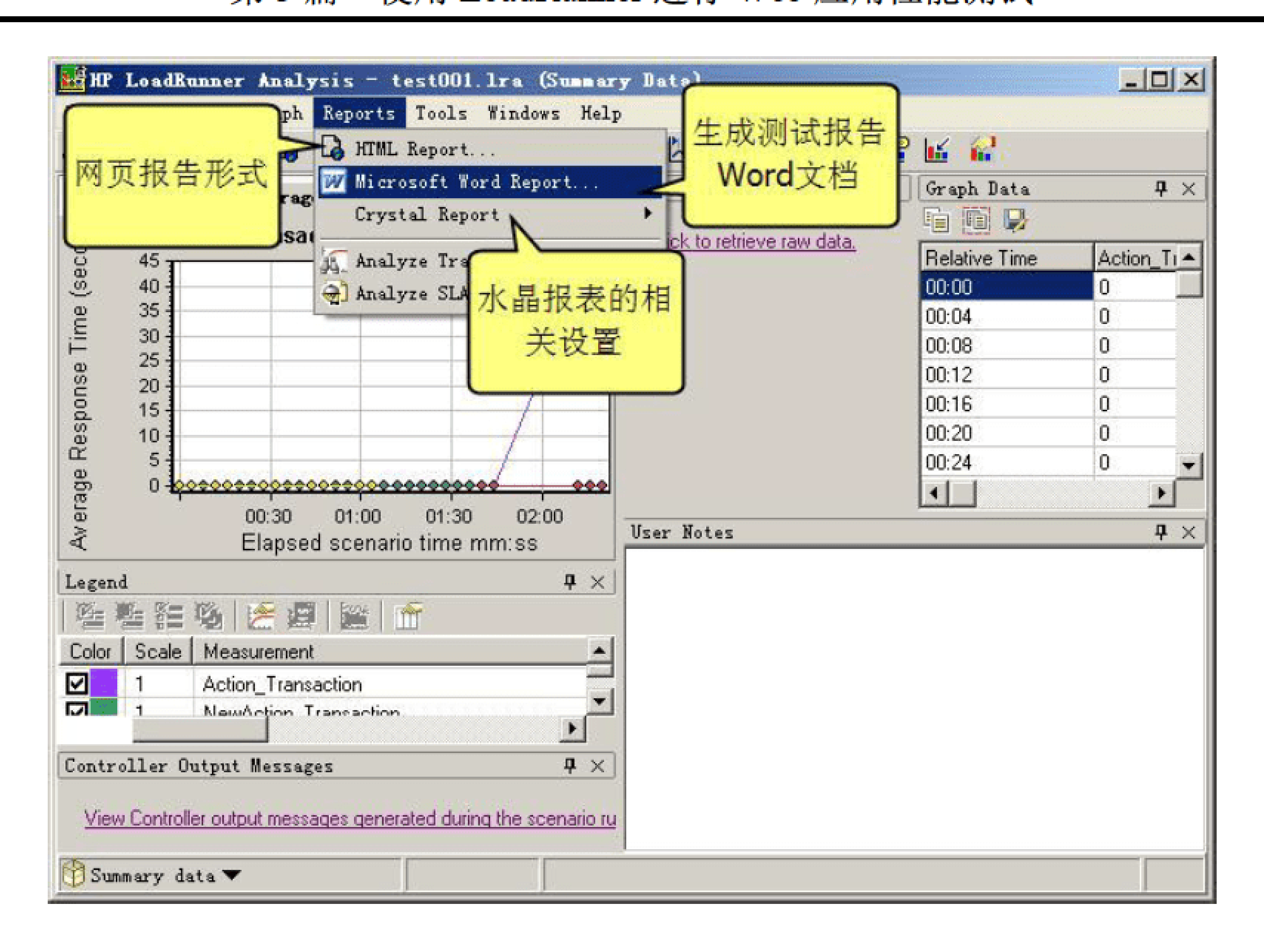


图 13-5　LoadRunner 的报告菜单

【网页报告格式】

对于网页报告（HTML Report）格式，LoadRunner 将弹出文件保存对话框，由用户指定位置和文件名称。经过短暂的等待，网页报告生成完毕后，在硬盘指定位置上将产生一个指定名称的网页、相关 CSS 文件、保存测试结果的 Access 数据库文件以及供用户下载保存图表的 Excel 文档。网页的内容则与第 12 章介绍过的分析概要一致。网页报告的好处是可以将其置于网站服务器的管理之下，相关人员输入网址即可看到，而不必各自在本地保留备份。

【水晶报表格式】

水晶报表（Crystal Reports）是一个商业智能软件，主要用于设计及产生报表。它可以通过使用本地或者 ODBC 的数据库客户端连接，能够访问各种不同类型的数据库数据，并以此来产生报告内容。LoadRunner 分析器为了方便客户，提供了水晶报表的支持，可以将报表导出为包括 Lotus Notes、Exchange 等多种格式的文件。关于水晶报表，由于不是本书的内容，感兴趣的读者可以阅读相关书籍。

最常用的测试报告格式可以说就是微软的 Word 格式了。它可以方便地被阅读、被发送。选择图 13-5 中的微软 Word 报告选项，弹出对话框如图 13-6 所示。图 13-6 中还有其他两个标签页，分别是 Primary Content（首要内容）和额外事项（Additional Items），都是对自动生成的报告内容进行删减的，如图 13-7 和图 13-8 所示。读者可以根据实际情况勾选或者取消其中的选项。

【不要忘记填写测试目的与结论】

不要忘记在图 13-7 中单击 Edit（编辑）按钮，并在弹出的窗口中输入测试目的和测试结论，这是需要测试工程师手工输入的，毕竟计算机在这方面不能代替人工的判断。

当所有设置都完成时，单击 OK 按钮，就可以得到一份很标准的测试报告了。

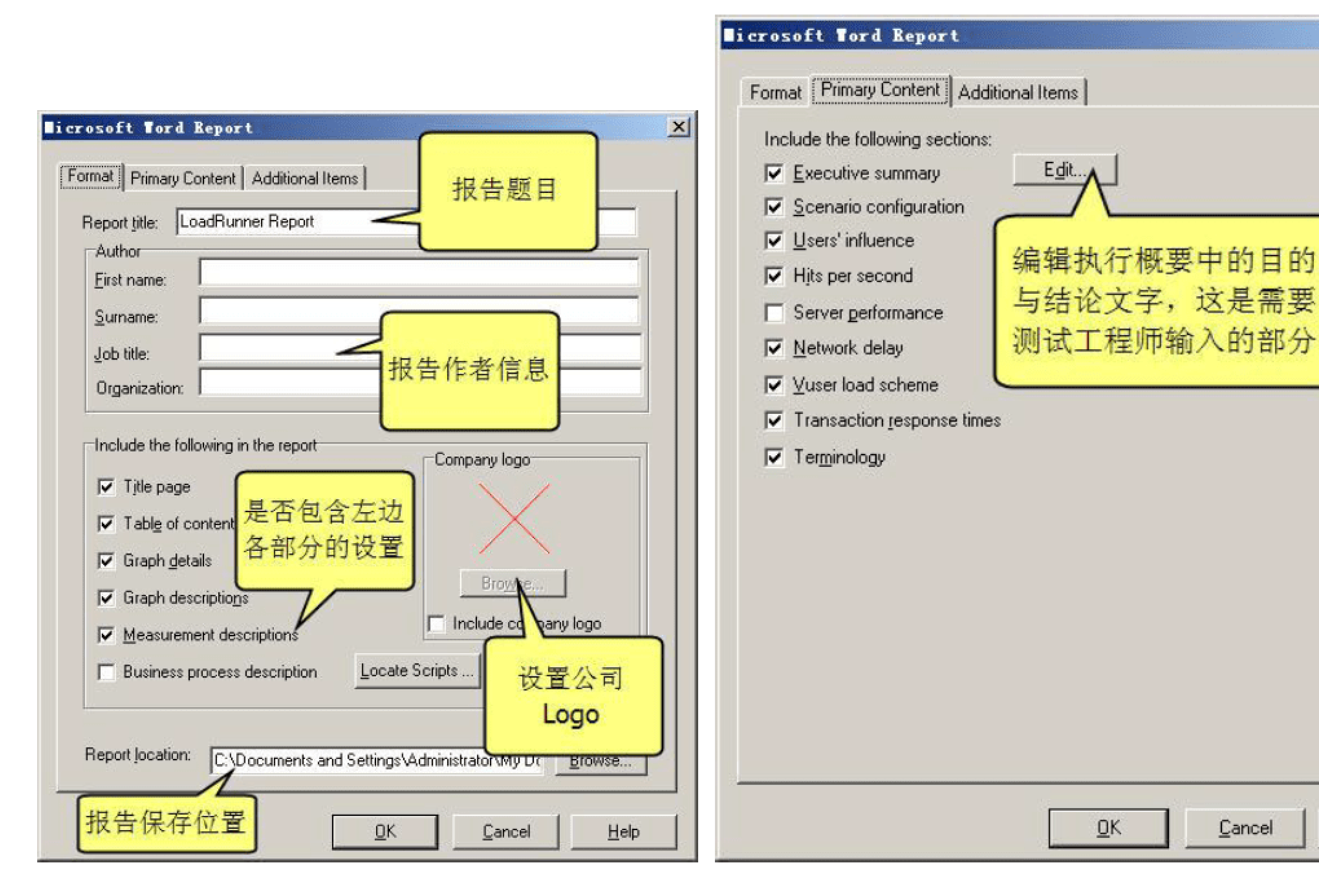


图 13-6　设置微软 Word 格式测试报告的对话框　　　　图 13-7　设置 Word 报告的首要内容

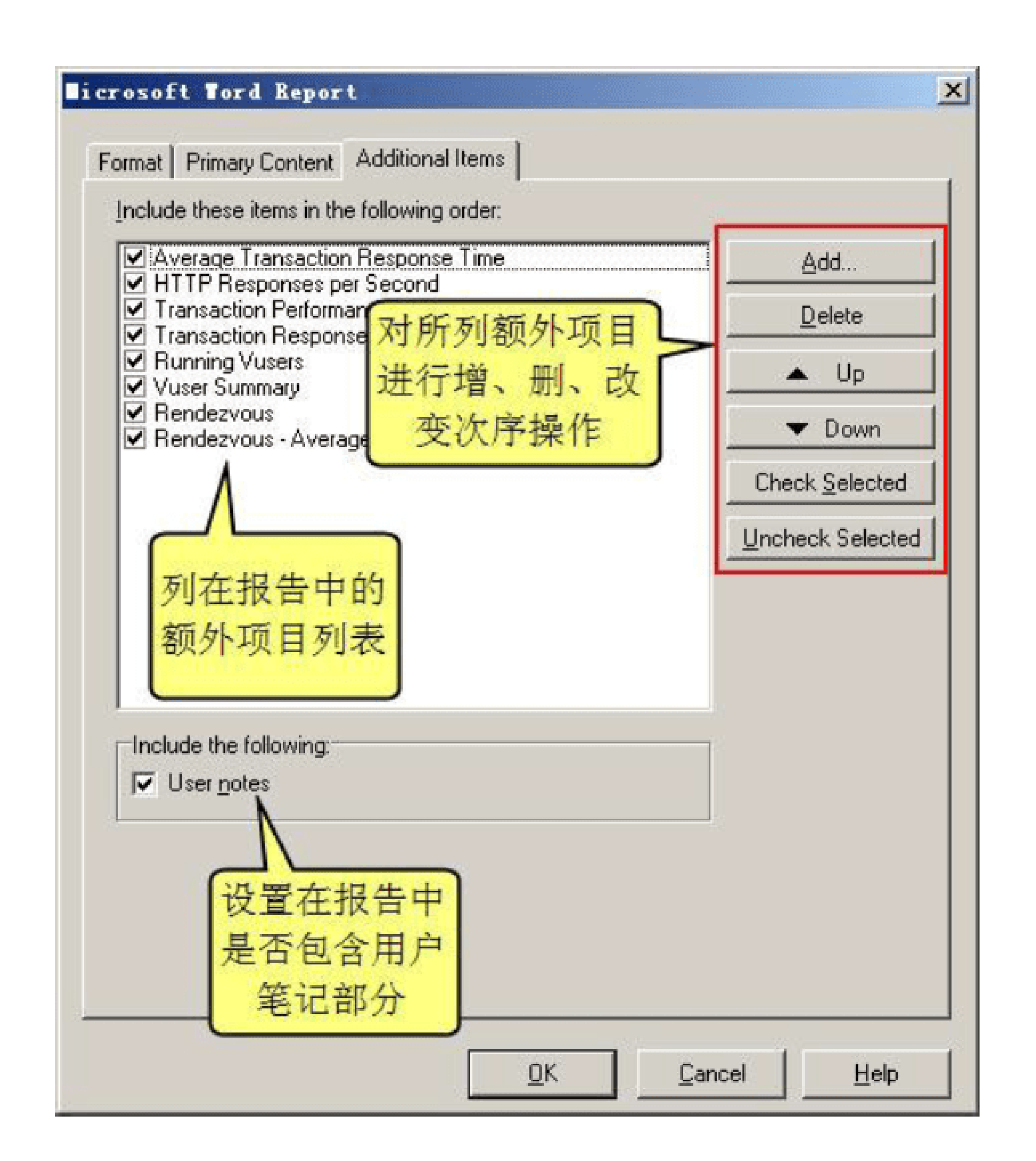


图 13-8　设置 Word 报告的额外项目

13.1.4　分析器导出数据

有的时候需要将 LoadRunner 得到的测试数据导出，在其他的工具软件中打开进行分析，这时，分析器的复制至剪贴板功能就很实用，如图 13-9 所示。

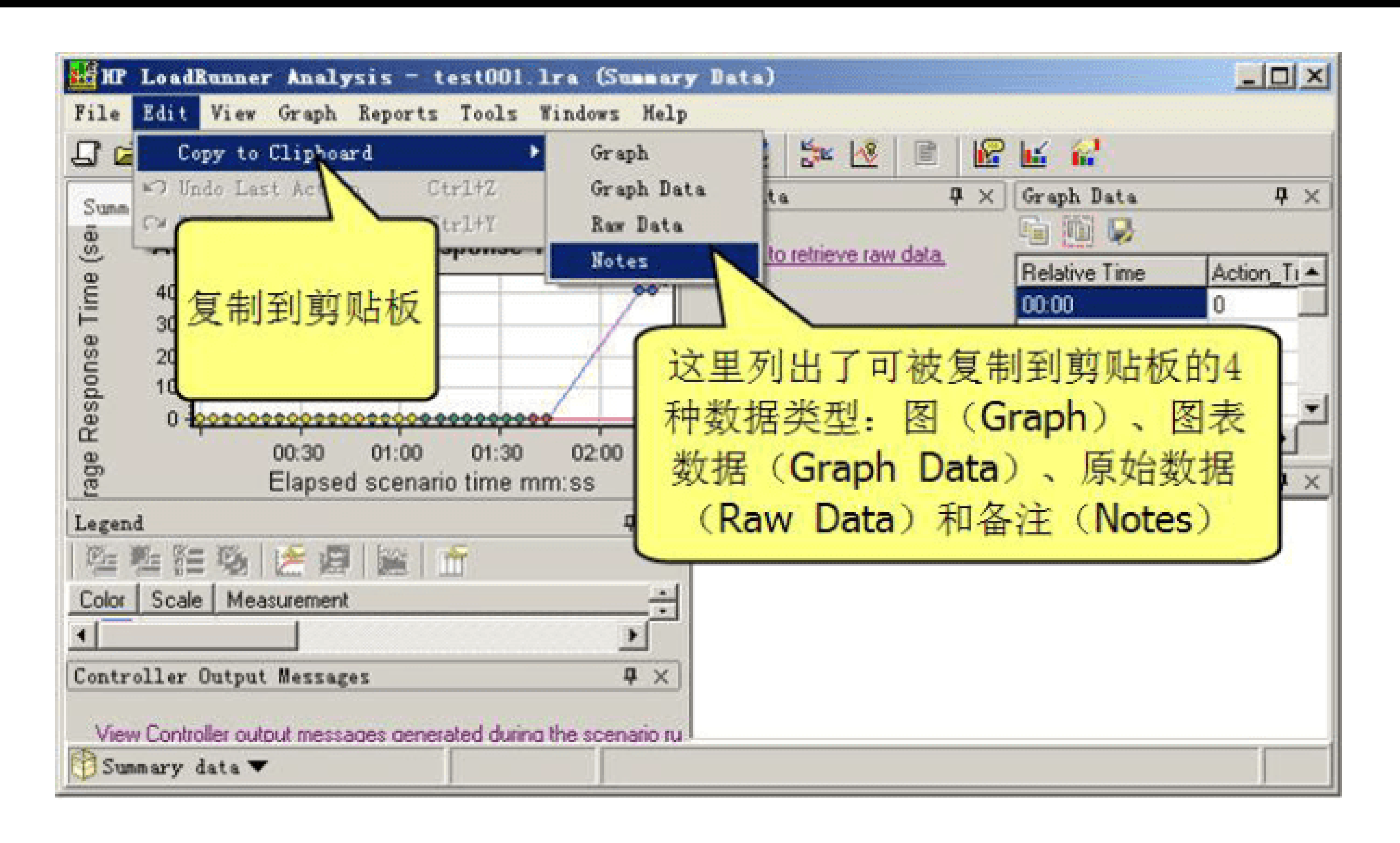


图 13-9　将几类数据复制至剪贴板的菜单

可以被复制的测试数据有如下 4 种：图表、图表数据、原始数据、用户笔记。

13.1.5　分析器数据存放位置

分析器数据存放的位置在某些情况下也是有必要知道的，比如需要把多次测试的结果复制到一台机器上进行统一分析的时候。分析器提供了相应的菜单来查看测试数据存放的位置，如图 13-10 所示。单击该菜单项后，弹出窗体如图 13-11 所示，可以根据图中提供的信息进行复制等进一步操作。

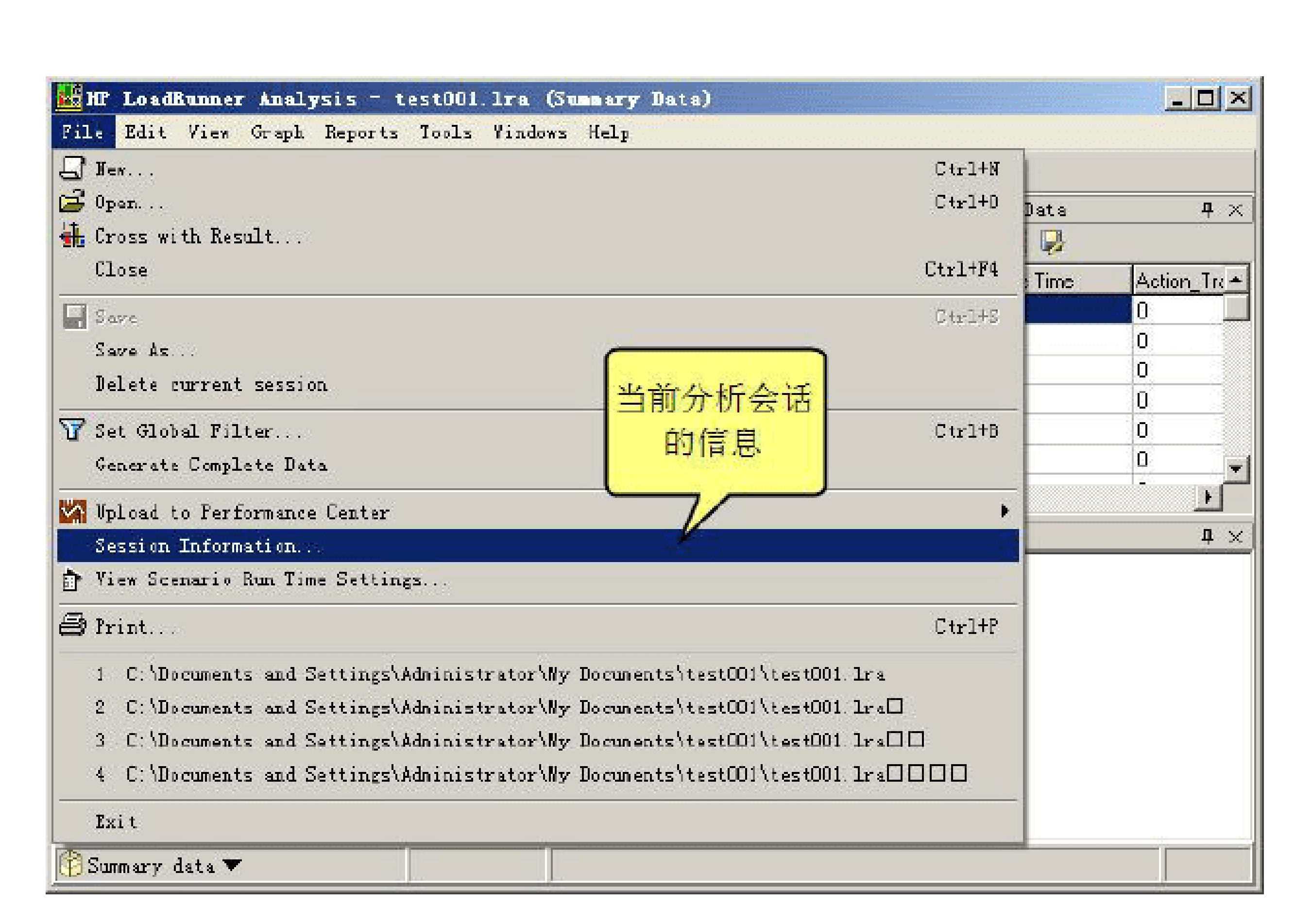


图 13-10　显示当前分析会话信息的菜单



图 13-11　显示当前分析会话信息的窗体

13.1.6　与其他工具软件协同

在分析器的工具（Tools）菜单中还列出了一些可以有利于协同工作的菜单项，如图 13-12 所示。如果在公司内部还装有 HP 的另外几种产品，比如 Performance Center 或者

Quality Center，则可以通过配置，将性能测试过程和结果加入到整个产品的测试过程与结果中去。感兴趣的读者可以阅读这两种软件的相关书籍。

另外，工具菜单中还有对外部监视器（External Monitor）、模板、Siebel 数据库调试、SAP 调试等的设置，可以在实际需要的时候参考相关文档进行学习。

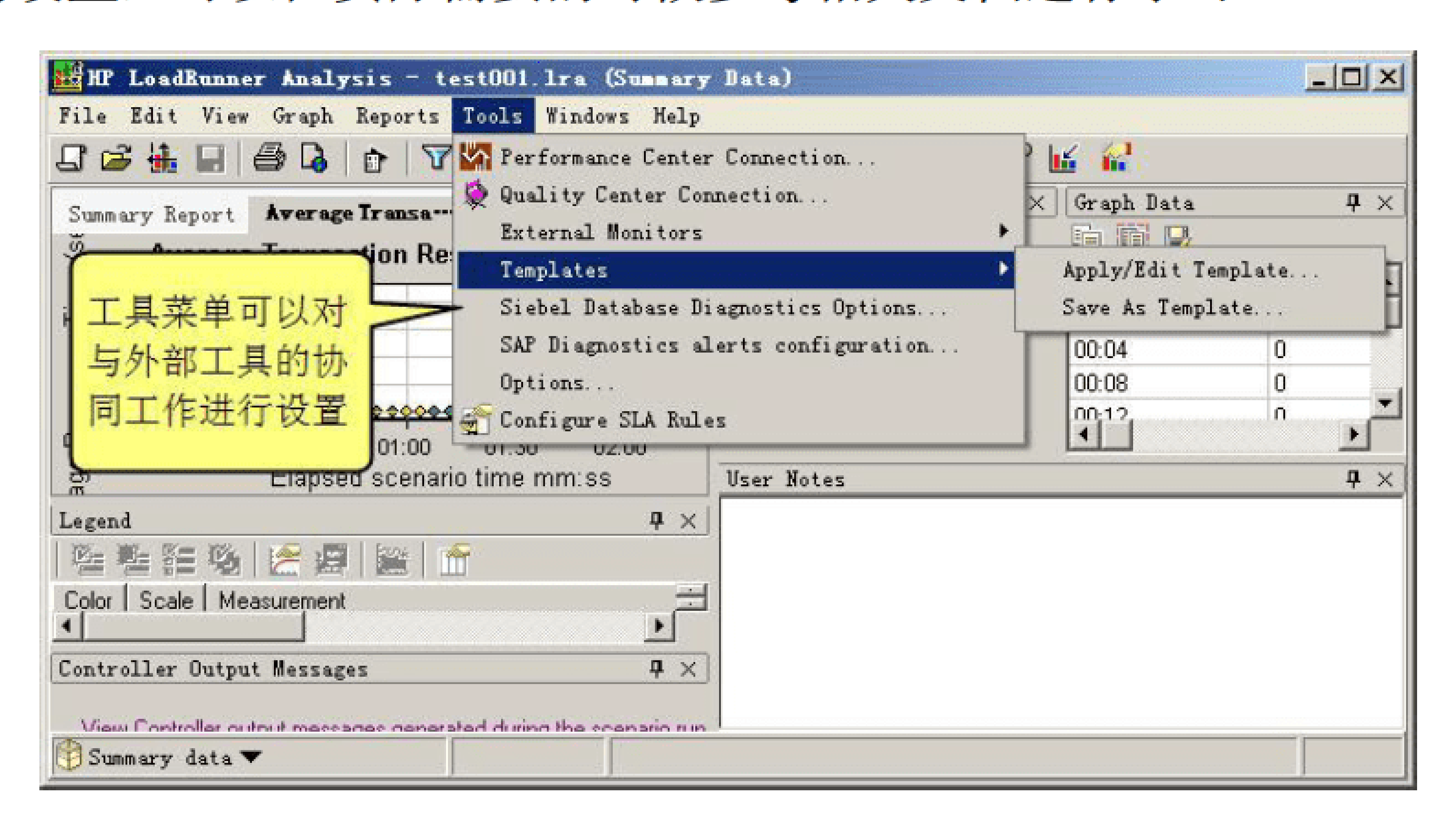


图 13-12　可设置与其他软件协同工作的工具菜单

13.1.7　分析器的全局设置（Options）

在介绍了几个分析器的使用技巧之后，有必要在本节的最后，对工具菜单中的分析器全局设置进行说明。全局设置所定义的属性将对所有用分析器打开的分析会话发生作用。选择 Tools|Options（工具|全局设置）选项，弹出设置窗体如图 13-13 所示。

全局设置窗体默认打开通用设置标签。

【事务百分比】

在概要报告（Summary Report）栏中，有两个数值值得注意，分别是事务百分比（Transaction Percentile）和显示最多某数值的失败事务，图中这两个数值分别为 90 和 5。事务百分比在最终的测试报告中将形成一个数据列，表示实际测试过程中有 90%的结果少于该列数值，即反映了 90%的数据处于什么程度；而报告最多将列出 5 条的失败事务作为参考。

- ❑ 切换至全局设置窗体的 Result Collection（结果集）标签，则可以设置分析器对于数据的处理方式，如图 13-14 所示。
- ❑ 所谓聚集显示，是指当数据来自某个分类的不同子分类时，是否自动产生一份将各子分类加和的数据。
- ❑ 切换至全局设置窗体的数据库（Database）标签，则可以设置测试数据保存于 Access 类型数据库还是 SQL Server 之中。
- ❑ 切换至全局设置窗体的网页调试（Web Page Diagnostics）标签，则可以设置对于动态网页数据是每个完全不同的 URL 单独计算还是以平均值来处理。
- ❑ 切换至全局设置窗体的分析事务（Analyze Transaction Settings）标签，则可以设置图表关联度的一些规则，以及事务报告中需要包含的图表类别。

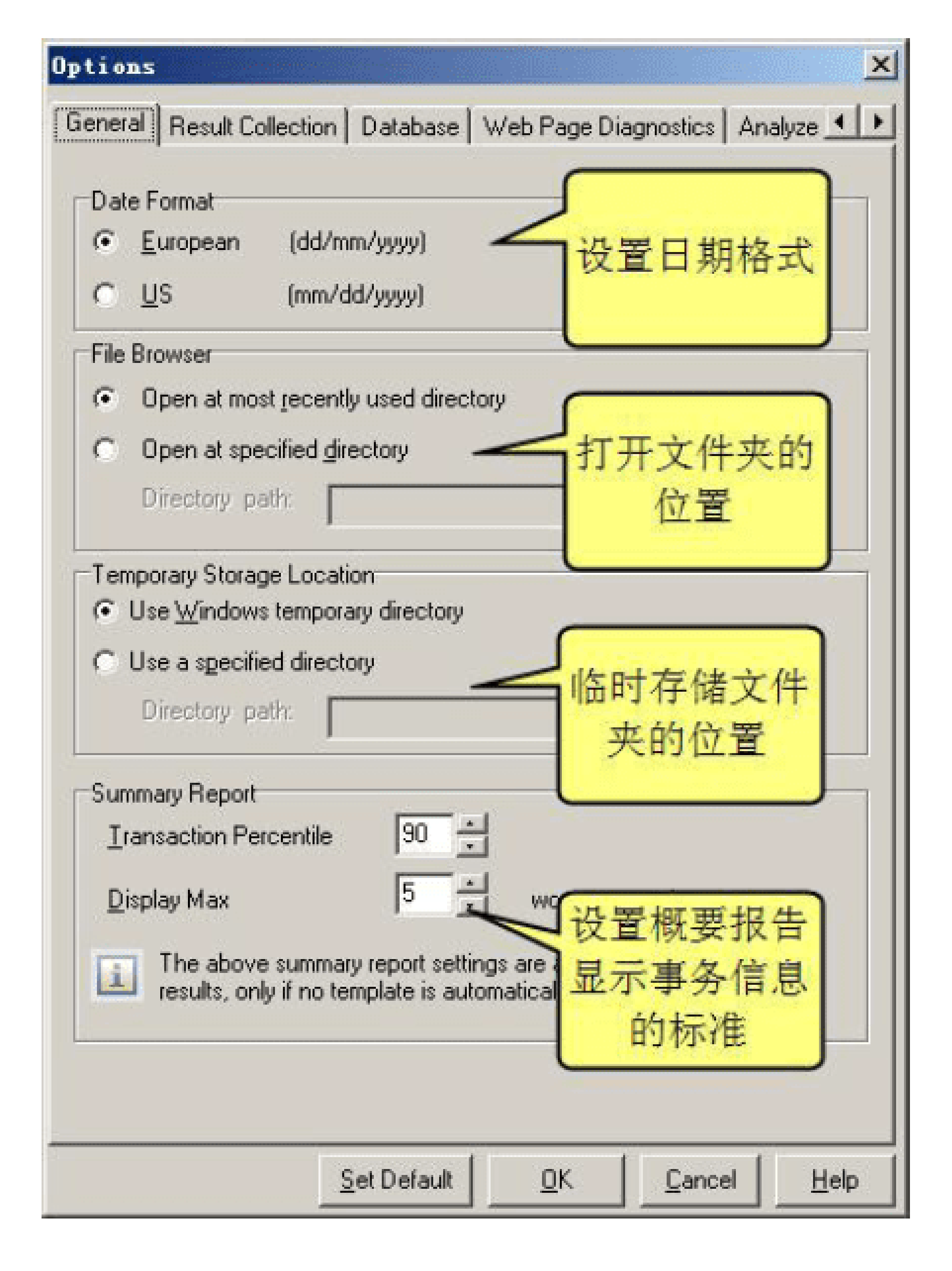



图 13-13　全局设置窗体：通用设置标签

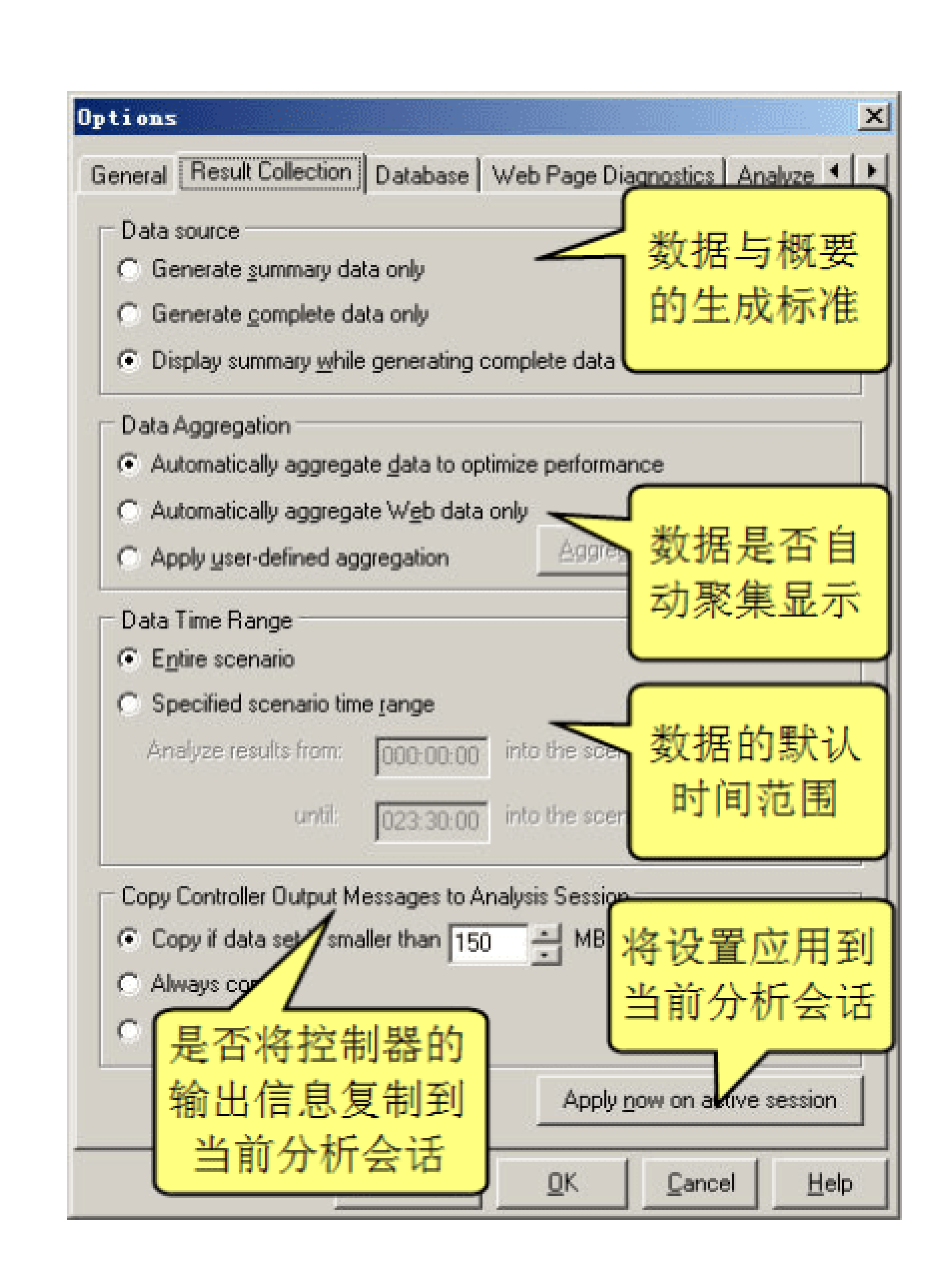


图 13-14　结果集设置

到本节为止，有关分析器的基本使用已经介绍完毕。在 13.2 节，我们将学习如何使用分析器的主要功能：即通过利用对图表的分析、图表的关联、交叉找出性能问题。

13.2　利用图表分析性能

图表比单纯的数据要更容易了解趋势、发现问题所在。在第 12 章中我们已经接触到分析概要中默认提供的图表，而在实际工作中，仅靠这几张图表是不够用的，因此有必要添加更多的图表。

13.2.1　添加更多图表

在分析器的图表（Graph）菜单下单击添加新图表菜单项，可以为当前的分析过程加入更多的图表，如图 13-15 所示。

添加新图表是在图 13-15 中显示的对话框中进行操作的。在窗口左边的图表列表中选择关注的某项，然后单击窗口右下方设置图表属性区域中的任一个属性，根据需要修改其值，完成后单击 Open Graph（打开图表）按钮就可以将所选择图表添加到分析中。

对图表属性进行设置的界面因图类型、属性不同而有所差异，图 13-16 显示了对绝大多数图表都具备的 Scenario Elapsed Time（场景执行时间）一项进行修改的界面。这项属性默认值是场景整个执行时间，可以修改为其中某一段的时间，这将影响图表 X 轴，即时间刻度的长度。

如果将图 13-15 中窗口左下方“只列出包含数据的图表类型”的默认选择去掉，列表中将增加更多种类的图表，可以很快明白这些图表和控制器中的图表列表是一致的。在当前场景执行结果中有数据的图表用蓝色字体标出。因此，场景中图表与分析中图表的关系

如图 13-17 所示。

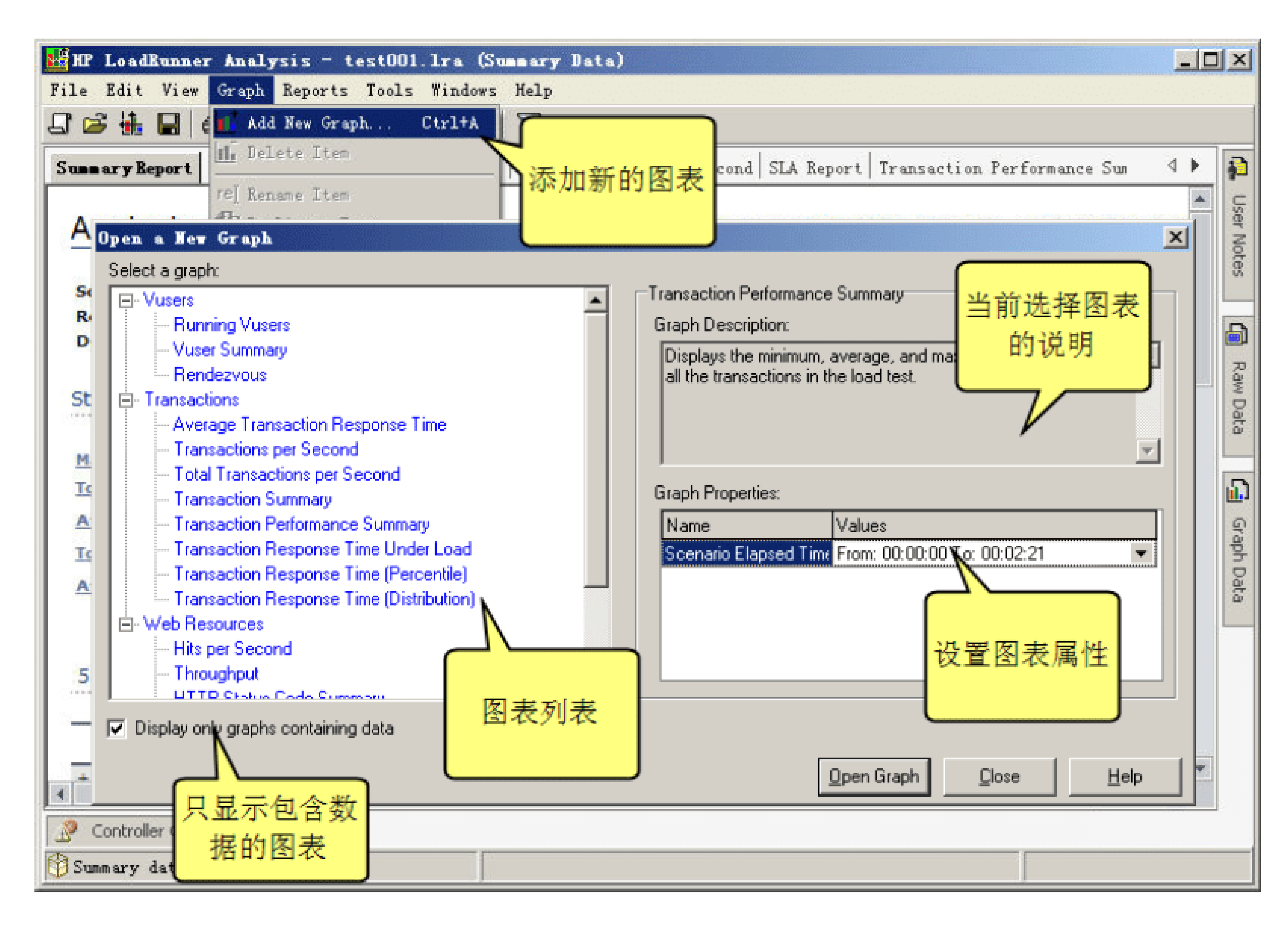


图 13-15　为当前分析添加更多图表

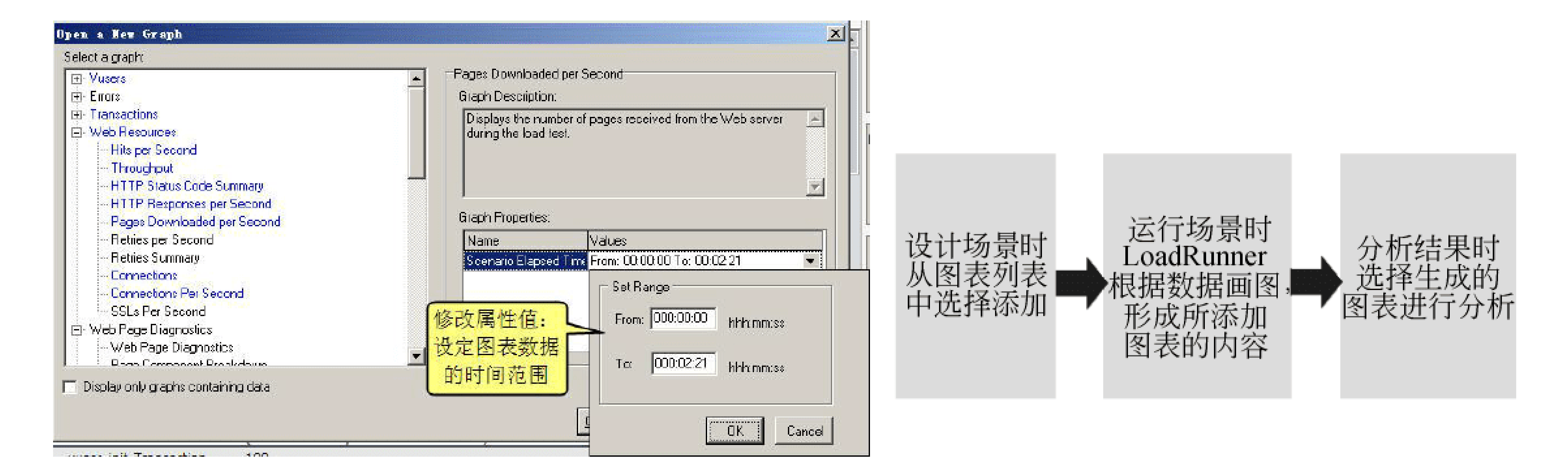


图 13-16　修改属性值界面示意　　　　图 13-17　图表在场景执行过程中的作用

根据实际工作的情况，可能需要重复多次添加新图表的操作直至完成。下面将讲解几种 Web 应用重要图表的分析。

13.2.2　虚拟用户图（Vuser 图）

分析器中的虚拟用户图，包含 3 张图表，分别为：

- ❑ 运行时虚拟用户图（Running Vusers）；
- ❑ 虚拟用户概要图（Vuser Summary）；
- ❑ 集合点图（Rendezvous）。

首先查看运行时虚拟用户图，如果该图没有显示或者添加，可使用 13.2.1 节的方法将其加入到当前分析会话中。

如图 13-18 是某次场景运行完毕后生成的运行时虚拟用户图。

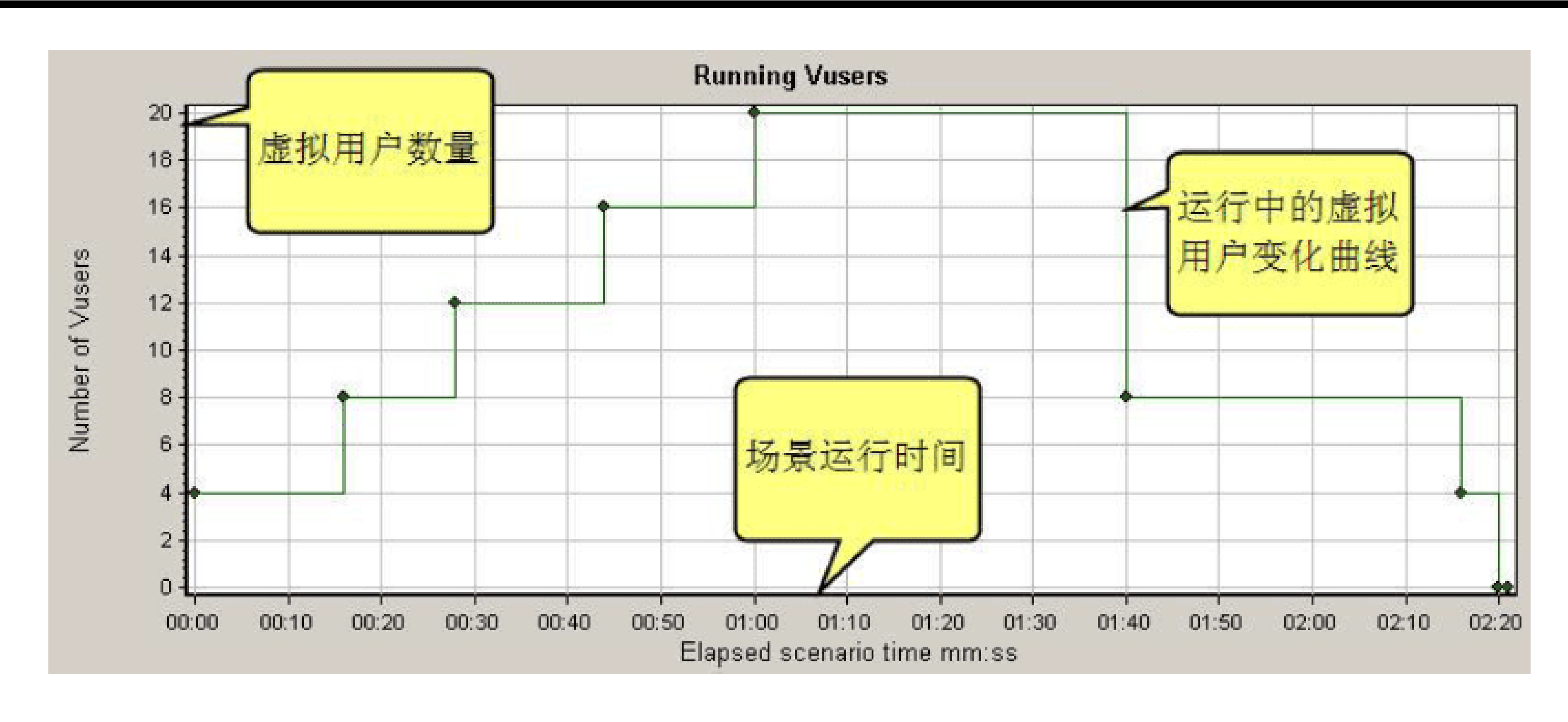


图 13-18　运行时虚拟用户图

运行时虚拟用户图表明了某一时刻参与场景运行的虚拟用户数量。从图 13-18 中可以发现，在场景运行初期，虚拟用户呈阶梯状逐渐增加，这是因为场景的设定是在 1 分钟的时间内，以渐进的方式达到 20 个虚拟用户。1 分钟之后，虚拟用户不再增加，而是维持在 20 个的水平。在 1 分 40 秒的时候，虚拟用户开始减少，直到最后完全停止，这正是场景停止退出执行的过程。

如图 13-19 显示了相同场景运行后的虚拟用户概要图，以说明虚拟用户组成情况，目前似乎看不到更多有用信息。如图 13-20 显示了某次场景运行后生成的集合点图。

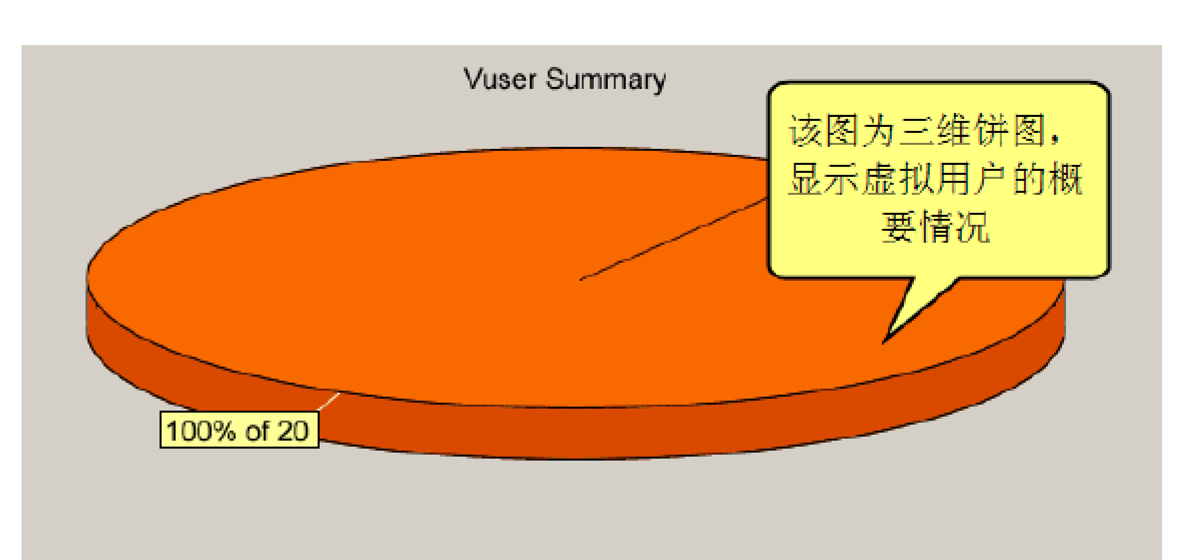


图 13-19　虚拟用户概要图

图 13-20　集合点图

集合点图的 X 轴表明场景开始运行后的时间，Y 轴表示 LoadRunner 从集合点处释放的虚拟用户数量。在图 13-20 中共有 4 个点，表明 X 轴标识的时间段内，共经历了 4 次集合点（虽然脚本中只有一个集合点的定义，但迭代了 4 次）。这 4 个集合点的数值分别为 4、8、10 和 10，最后 2 个数值是符合场景中集合点设置的（即当有 10 个虚拟用户同时到达时，即可开始释放，以同时进行下一步的操作）。那么，为什么第 1、2 次没有达到 10 的数值也开始释放了呢？

原因是集合点释放的另一个设置：超时。默认情况下，当集合点处现有最后一个到达与下一个到达虚拟用户之间时间相差 30 秒的情况下，LoadRunner 也会开始释放用户，进行下一步的操作。从图 13-20 中可以发现，在场景运行初期，有一些用户启动较慢，因此没有“跟上大部队”，导致前面的两个集合点都没有聚集 10 个用户就开始释放了。随着场景运行的进行，后几次迭代中用户已经不需要初始化了，因此相邻用户之间不会超时，从而保证了集合点处用户数量的要求。

虚拟用户图反映了用户数量这一性能测试中的重要因素，当需要观察性能测试结果与用户数量增加有何关系时，可以使用这类图表。

13.2.3　细化图表数据：过滤/分组

在进一步介绍更多的性能图表之前，有必要介绍细化图表数据的方法。在 13.2.2 节的用户概要图（图 13-19）中我们发现，整个图的内容只是一个数值，没有太多意义。是否这就是 LoadRunner 对于用户概要所能够提供的所有信息呢？

回答是否定的。为了展示更多的信息，我们需要使用过滤/分组功能、下钻（Drill down）功能细化图表数据，本小节先介绍前者。LoadRunner 的“设置过滤与分组”（Set Filter/Group by…）命令提供了分类显示图表数据的功能。通过在图表空白处右击，在弹出的快捷菜单中，或者单击分析器的 View（查看）菜单，都可以找到以上菜单项，如图 13-21 所示。

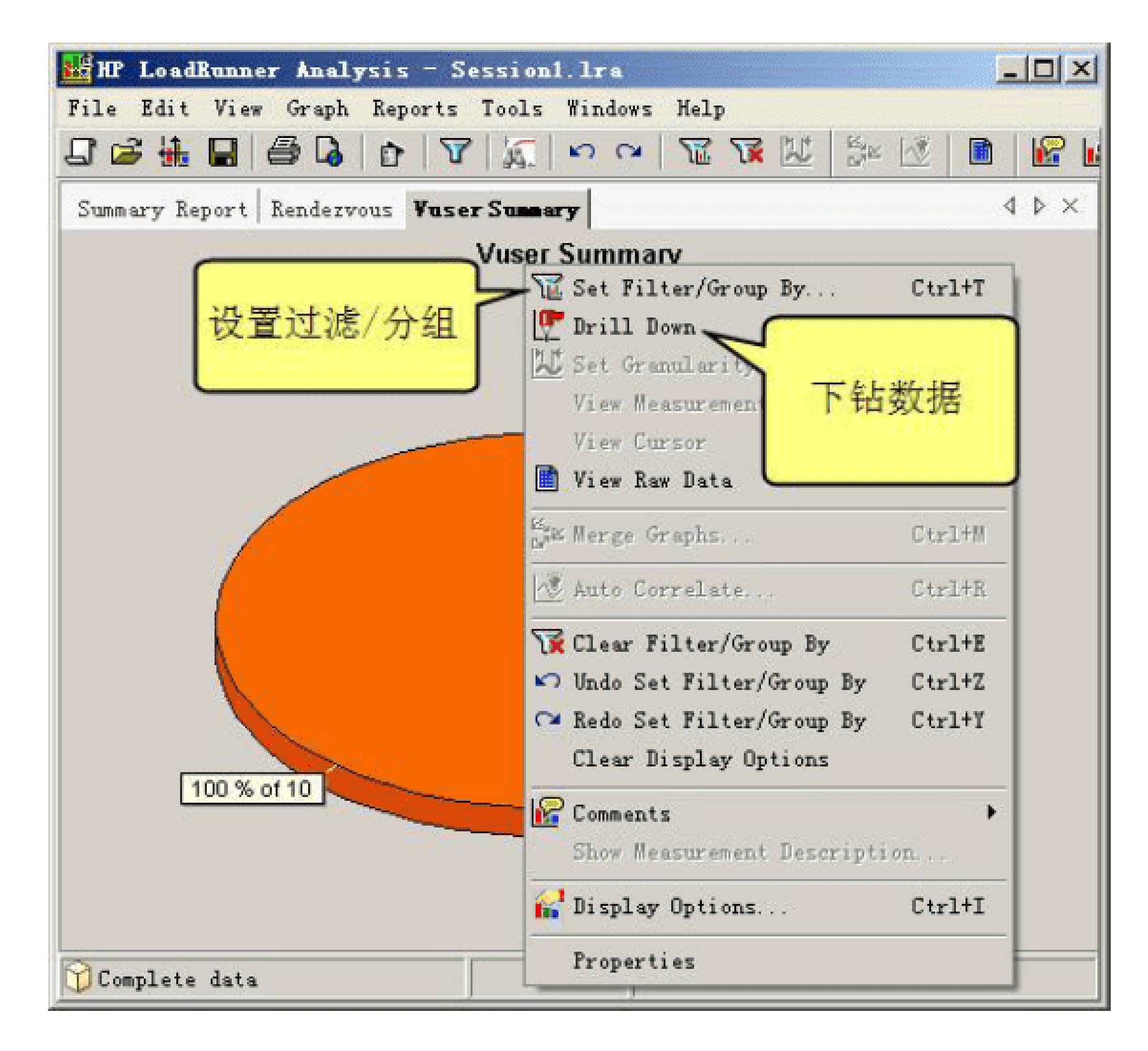


图 13-21　在图表空白处右击，在弹出的菜单中可对图表显示数据进行细化

单击设置过滤/分组，弹出窗体如图 13-22 所示。窗体分为过滤器设置和分组设置两部分。过滤器列表举出当前图表可用的过滤器，在图中是虚拟用户结束状态、场景运行时间以及虚拟用户 ID 这 3 种。每一种过滤器都有多种状态，利用判断条件（Criteria），可以筛选符合条件（比如虚拟用户结束状态等于失败这样的条件）的数据。

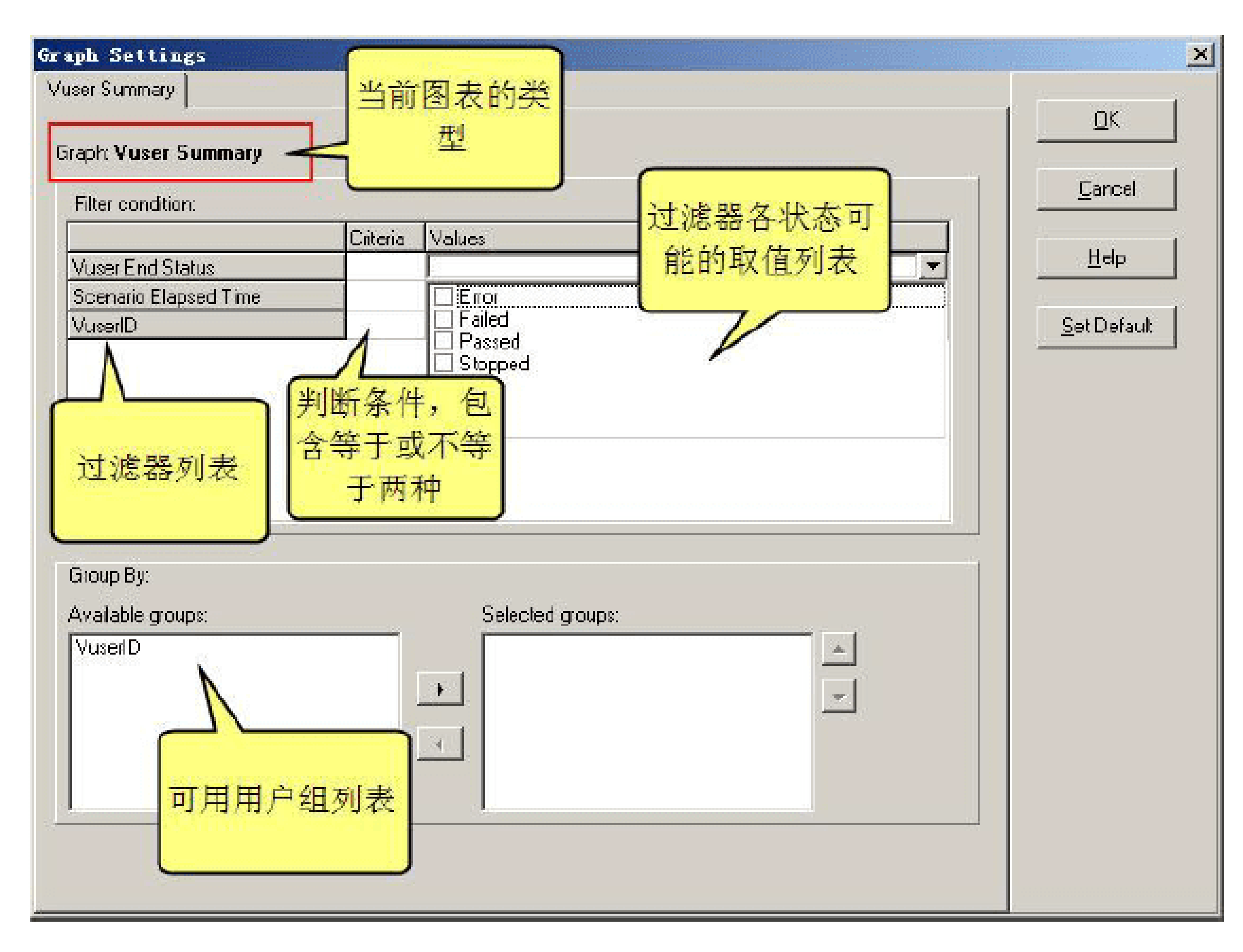


图 13-22　为图表设定过滤/分组条件

在图 13-22 所示窗体的下方是分组设置，只需要选择某个可用的组添加进右边的列表框中即可。设置完毕后，单击 OK 按钮即可生效，图 13-21 将变成包含分类数据的饼图，如图 13-23 所示。

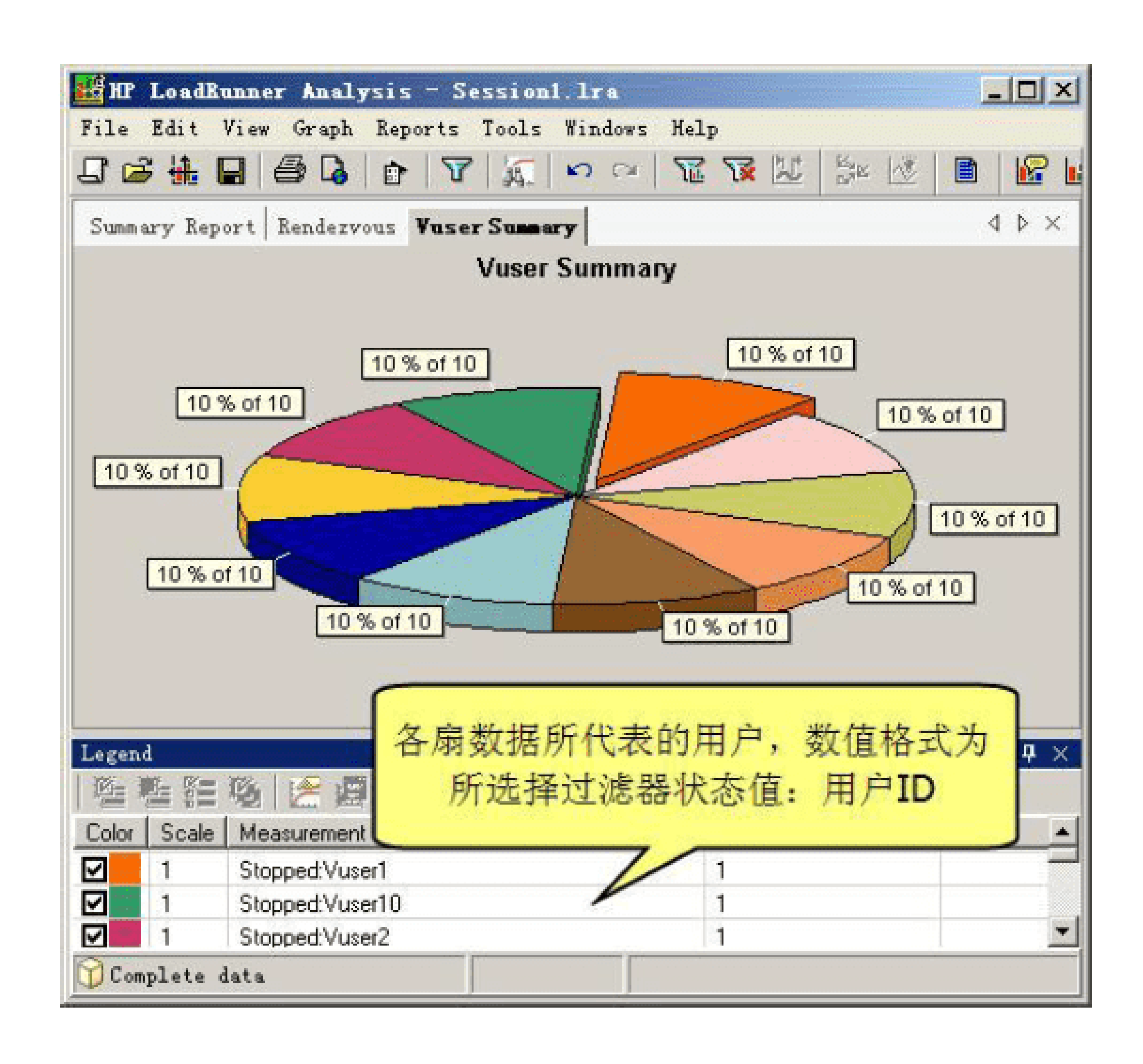


图 13-23　应用过滤器和分组之后图表能显示更细化的数据

【过滤分组的适用范围】

对于所有类型的图表，都可以通过应用过滤/分组来进一步显示细化数据。

13.2.4　细化图表数据：下钻

除了过滤/分组，还可以使用下钻来查看图表包含的详细数据。

【下钻】

所谓下钻，通俗地理解就是从已有的数据中挖掘更细分类的数据。比如已知北京市人口为 1500 万，那么其中各区县的人口是多少呢？这个获取各区县人口数的过程就可以被认为是下钻。当然，由于下钻是数据挖掘的一个术语，有更严格的定义，如果读者感兴趣，可以查阅专门的书籍。

在图 13-21 的空白处右击，在弹出的快捷菜单中选择 Drill down（下钻）命令，将出现下钻设置窗口，如图 13-24 所示。

在可供下钻的数据类型中进行选择，然后单击 OK 按钮生效，图 13-21 将变为图 13-25 的形式。可以发现，图 13-25 与应用过滤/分组后的图 13-23 并没有区别。这是由于对同一份原始数据，应用同样的分组（无论采取过滤/分组还是下钻）所得到的数据肯定是一致的。

【过滤分组与下钻】

实际上，过滤/分组更多地被归为查看图表细化数据的方法，而下钻则被归类于分析图表数据的方法，设置窗口的操作也相对简便。

【下钻的适用范围】

对于 Web Page 调试图表分类中的 Web Page 调试图，是无法使用下钻的，右键菜单中

并无下钻菜单项。其他的图表我们可以放心地使用。



图 13-24　下钻设置窗口

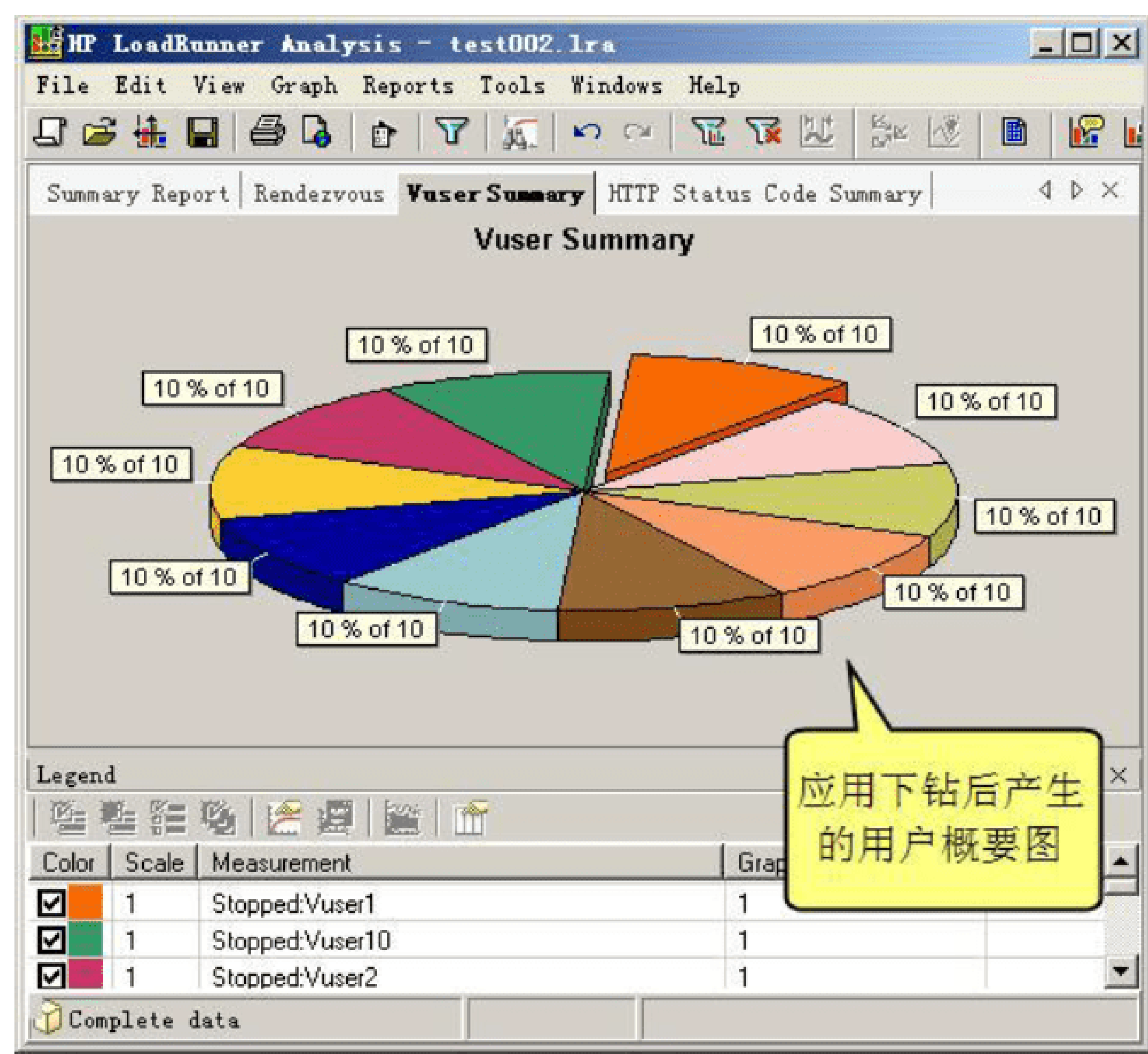


图 13-25　对用户概要图进行下钻

13.2.5　细化图表数据：取消过滤/分组/下钻设置

如果对某图表应用过滤/分组或者下钻之后，想取消或者改变分组设置，则可以采取如下两种操作：

（1）对于改变分组设置的情况，可以重新右击过滤/分组或者下钻菜单，在设置窗体中修改分组信息确定即可。

（2）对于取消过滤/分组和下钻、需要将现有图表恢复到最初状态的情况，则可以在图表空白处右击，在弹出的快捷菜单中选择 Clear Filter/Group by（清除过滤/分组）命令即可。

【下钻的取消】

需要特别注意的是，下钻的取消并没有专门的菜单项，依然是通过清除过滤/分组来实现的。

13.2.6　辅助图表工具：设置粒度

LoadRunner 为了测试工程师读图的方便，右击图表后在弹出的快捷菜单中还增加了几个菜单项，分别能够完成如下的功能。

- ❑ 设置粒度（Set granularity）：用于修改图表数据粒度，便于发现数据的宏观趋势。
- ❑ 查看度量趋势（View Measurements Trends）：通过计算图表中当前数值与前值的差，便于发现线图中各曲线的趋势。
- ❑ 显示光标（View Cursor）：将在图表中显示十字交叉线，两线交点就是当前光标

位置，便于在复杂图表中定位当前数据点。

本节我们先来学习设置粒度的方法，将以图 13-20 所示的集合点图为例。另外，查看度量趋势的使用讲解将在介绍更多的图表类型之后进行。

【粒度】

粒度在 LoadRunner 中可以简单地理解为相邻数据点的度量间隔。在图 13-20 中，每个数据点间隔大致为 40 秒，因此 40 秒就是该图原始的粒度。

修改粒度是为了发现图表中数据的长期趋势，因此设置的粒度一般都比原始粒度要大（设置的粒度比原始粒度小，但数据之间的度量间隔还是不变的，所以没有意义）。

在图 13-20 的空白处右击，在弹出的快捷菜单中选择 Set Granularity（设置粒度）选项，会出现设置窗口，如图 13-26 所示。LoadRunner 为了适用所有的图表，窗口中默认的初始粒度是 4 秒。

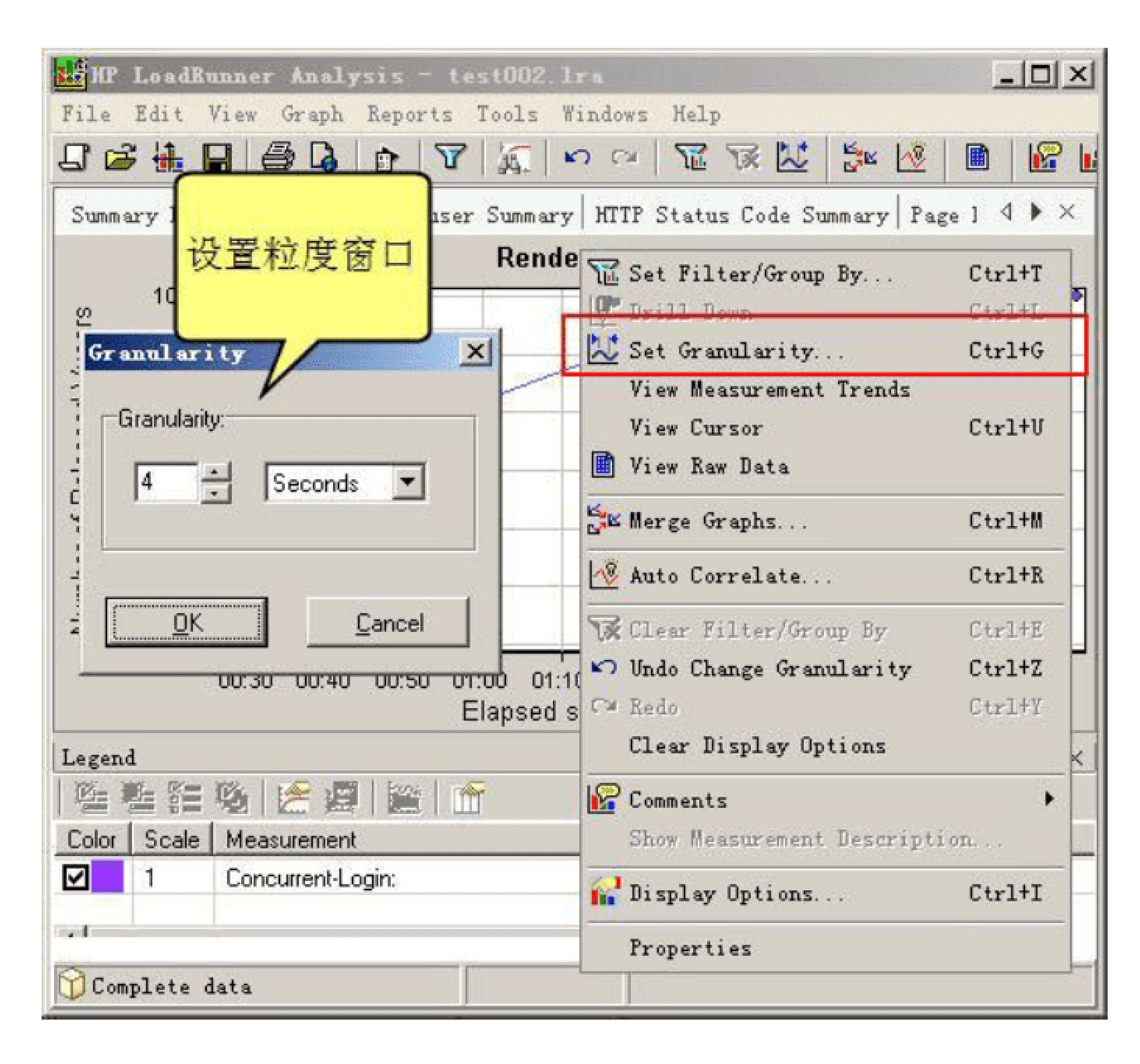


图 13-26　设置图表的粒度

实际的粒度要根据当前图表的原始粒度而定，由于图 13-20 中数据的原始粒度为 40 秒，因此我们通过图 13-26 中的下拉列表框，将粒度修改为大于 40 秒的数值，比如 1 分钟。设置完毕后，单击 OK 按钮使设置生效。

为了说明问题，图 13-27 将设置前的集合点图（即图 13-20）与设置后的集合点图从上至下并列在一起，从中可以发现下半部分修改粒度后的图中数据点变少了（因为粒度 1 分钟之内的数据被忽略），但是能够去除上半部分原图中前 3 个点的小变化趋势，而总体大趋势不变。

如果需要将图 13-27 中下半部分的图表恢复到上半部分，可以通过在图表空白处右击，在弹出的快捷菜单中选择 Undo Set Granularity（撤销设置粒度）选项即可，如图 13-28 所示。

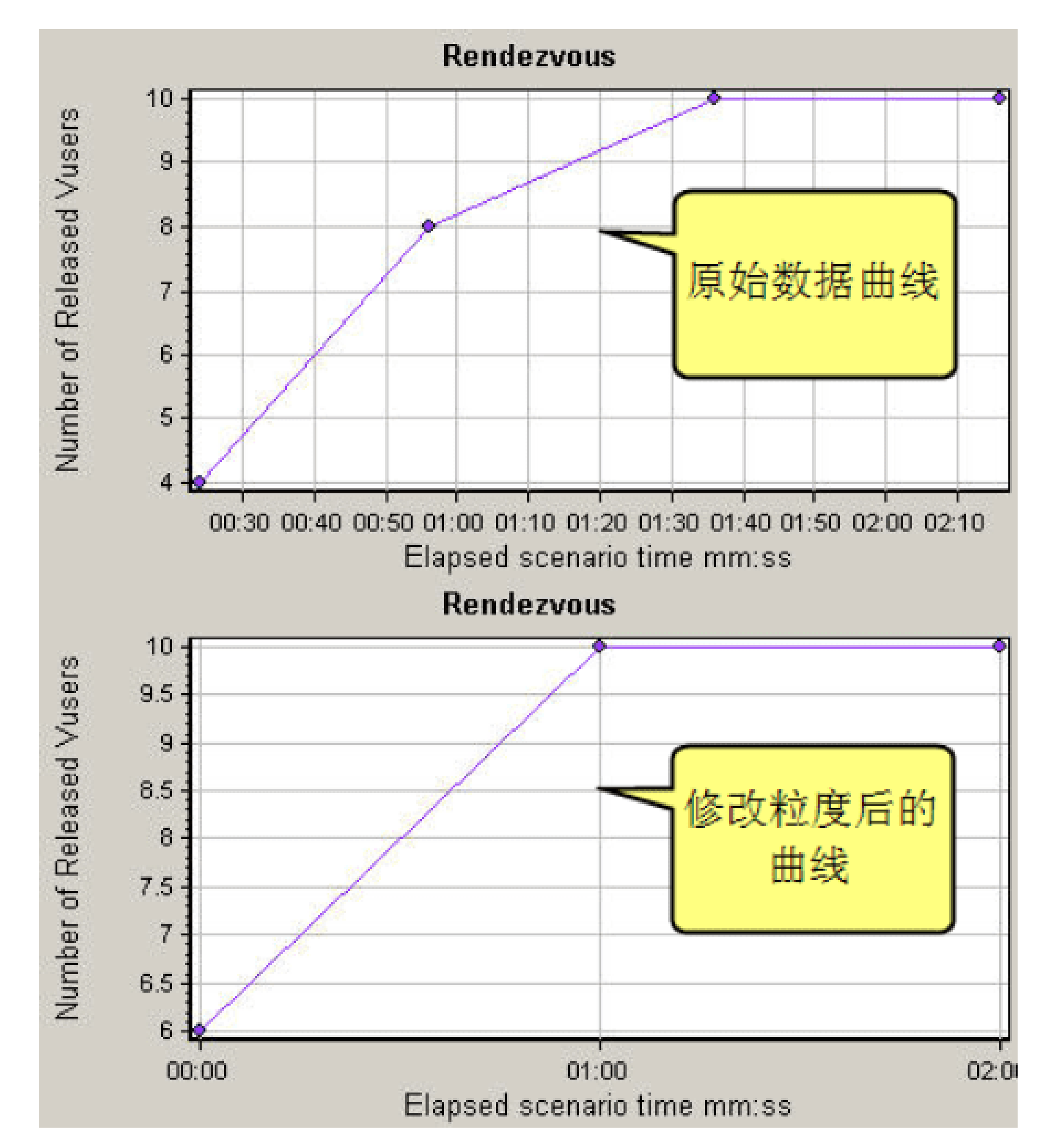


图 13-27　设置粒度前后集合点图的变化

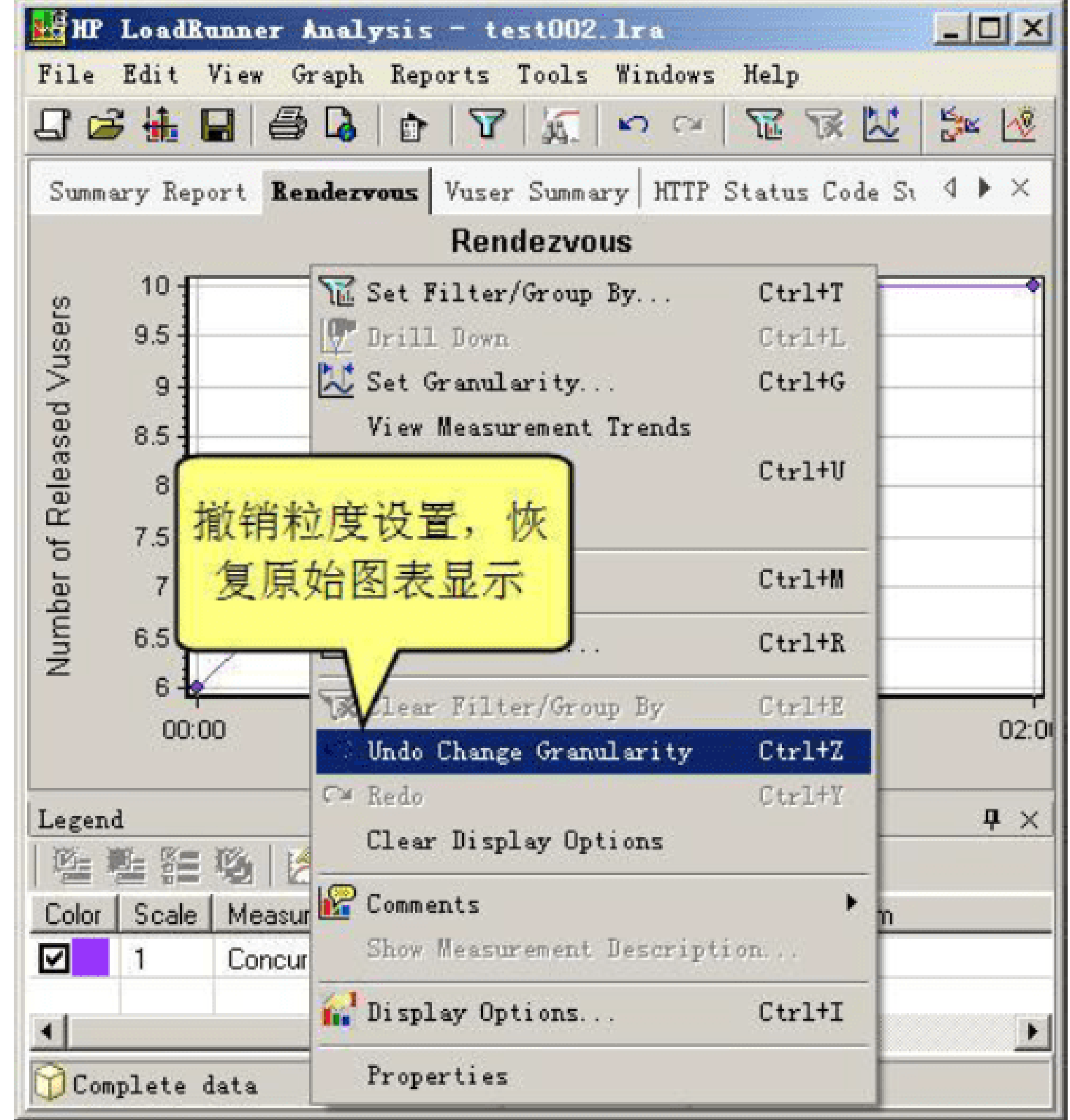


图 13-28　撤销粒度设置

13.2.7　图表辅助工具：显示光标

屏幕上的光标（Cursor）显示了当前鼠标或者文字的位置，在 LoadRunner 的分析器界面，如果图表中内容较多，查看会比较困难，使用光标是不错的办法。

在任一图表空白处右击，在弹出的快捷菜单中选择 View Cursor（显示光标）选项，在图中将显示纵横线，它们的交点代表鼠标当前的位置。当光标处于图表数据曲线上的数据点时，鼠标将变成手的形状，并有该点的提示信息出现，如图 13-29 所示。

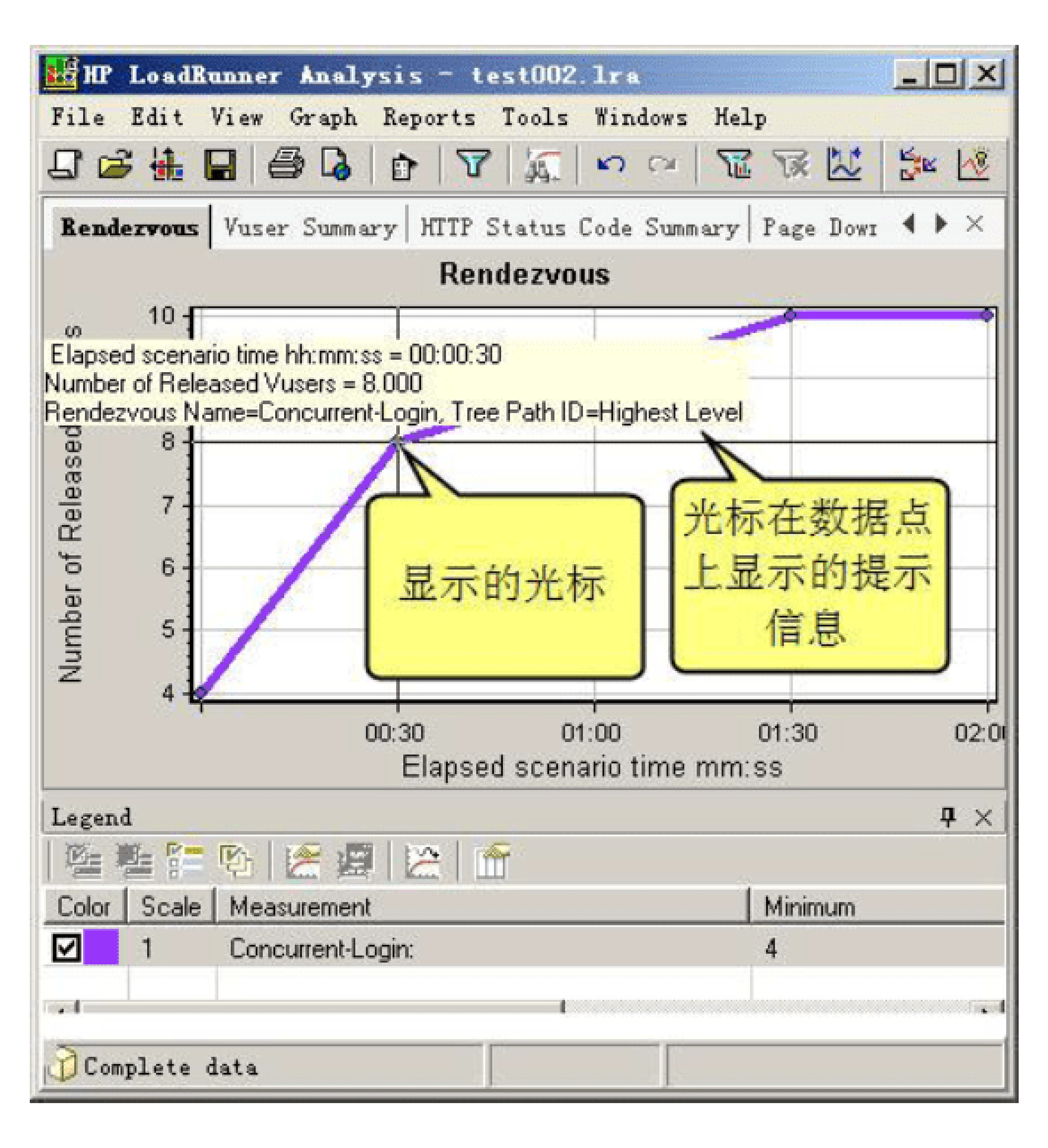


图 13-29　显示光标并查看数据点信息

13.2.8　事务图（Transaction 图）

在分析器中可以通过增加事务图来分析事务的运行情况。事务图包括如下图表：

- 平均事务响应时间图（Average Transaction Response Time）；
- 每秒事务数量图（Transaction Per Second）；
- 每秒事务总数量图（Total Transactions Per Second）；
- 事务概要图（Transaction Summary）；
- 事务性能概要图（Transaction Performance Summary）；
- 负载下事务响应时间图（Transaction Response Time Under Load）；
- 事务响应时间百分比图（Transaction Response Time (Percentile)）；
- 事务响应时间分布图（Transaction Response Time (Distribution)）。

下面的章节将一一介绍。

13.2.9　平均事务响应时间图

与其他大多数分析器图表不同，在增加平均事务响应时间图之前，还需要设置图表是否需要包括思考时间：包含与否会造成图表中的数值不同，如图 13-30 所示。单击 Open Graph（打开图表）按钮，平均事务响应时间将添加到当前分析会话中，如图 13-31 所示。

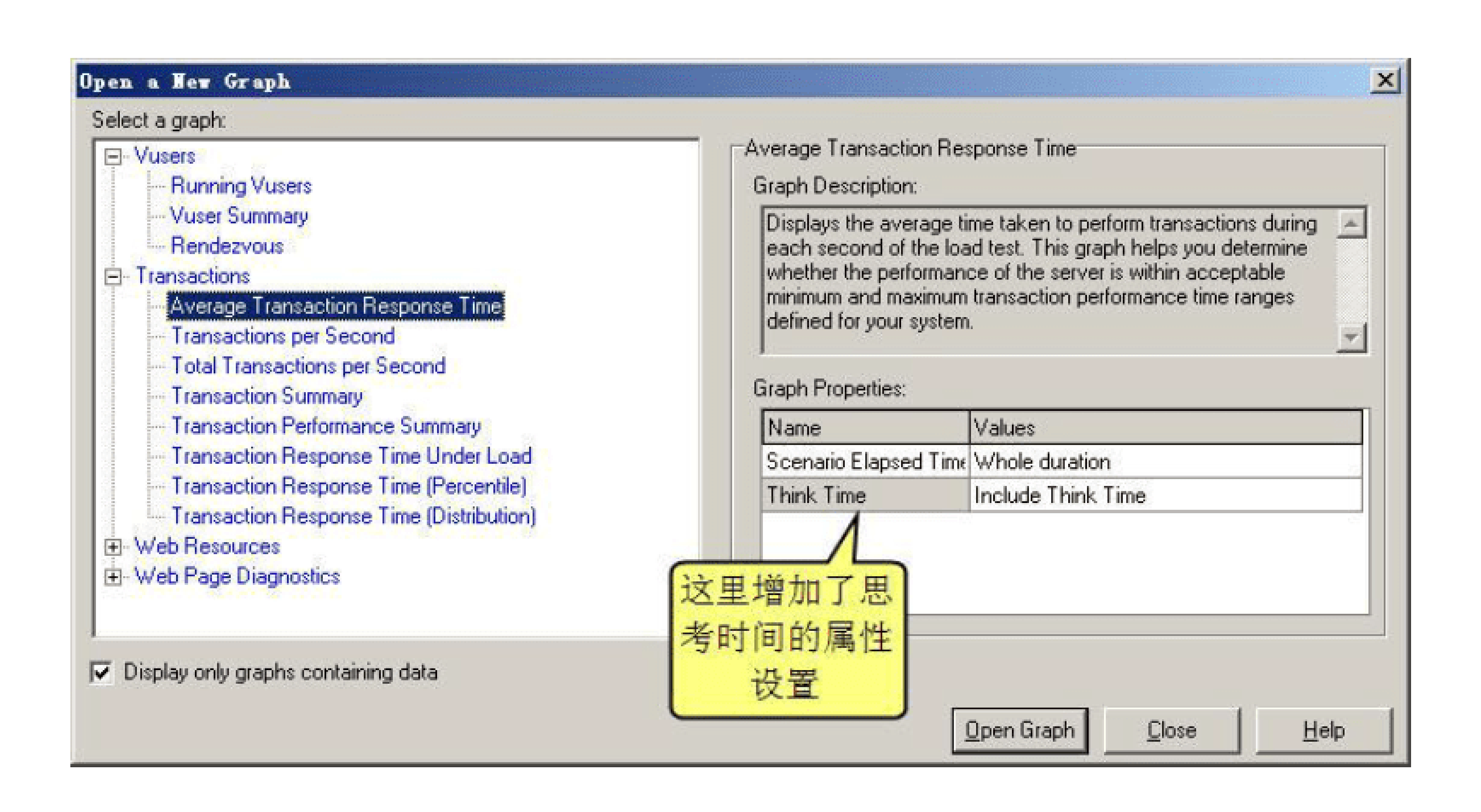


图 13-30　增加平均事务响应时间图前的属性设置

平均事务响应时间图的 X 轴表示场景执行的时间，Y 轴表示每个事务按秒计算的平均响应时间。从图 13-31 中可以看到，由于所选择脚本比较简单，只有一个真正用于测试的事务（Action_Transaction），因此图中包含有意义数值的事务曲线只有一条，并用不同的颜色和别的事务曲线区分。

在图 13-31 下方还有各事务平均响应时间的统计数值，可以方便地勾选以显示或者隐藏某条事务曲线，这些操作在第 12 章都已经提到。

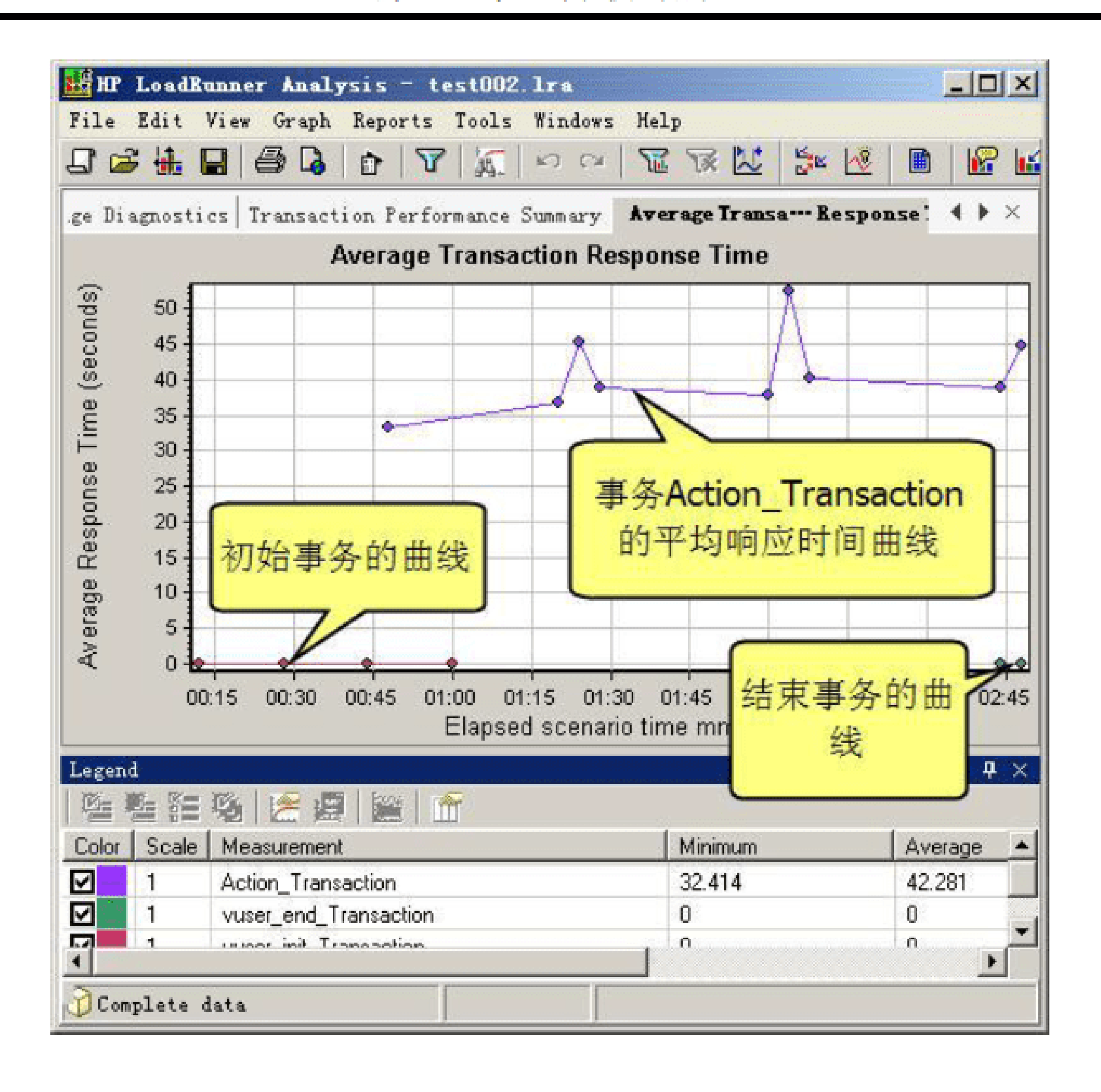


图 13-31　平均事务响应时间图

在 Action_Transaction 事务曲线上右击，随后弹出的快捷菜单中除了之前介绍的设置粒度、查看光标等命令之外，还有事务图特有的 Show Transaction Breakdown Tree（显示事务详细内容树）以及当前事务的 Web Page Diagnostics for “Action_Transaction”（网页调试图）命令，后者将在本章的 13.2.12 节中介绍。

【菜单的小差别】

如果不在图 13-32 中的某条事务曲线上右击，而只在图表空白处右击，则弹出的快捷菜单中“当前事务的网页调试图”是不会出现的，在此特别说明。

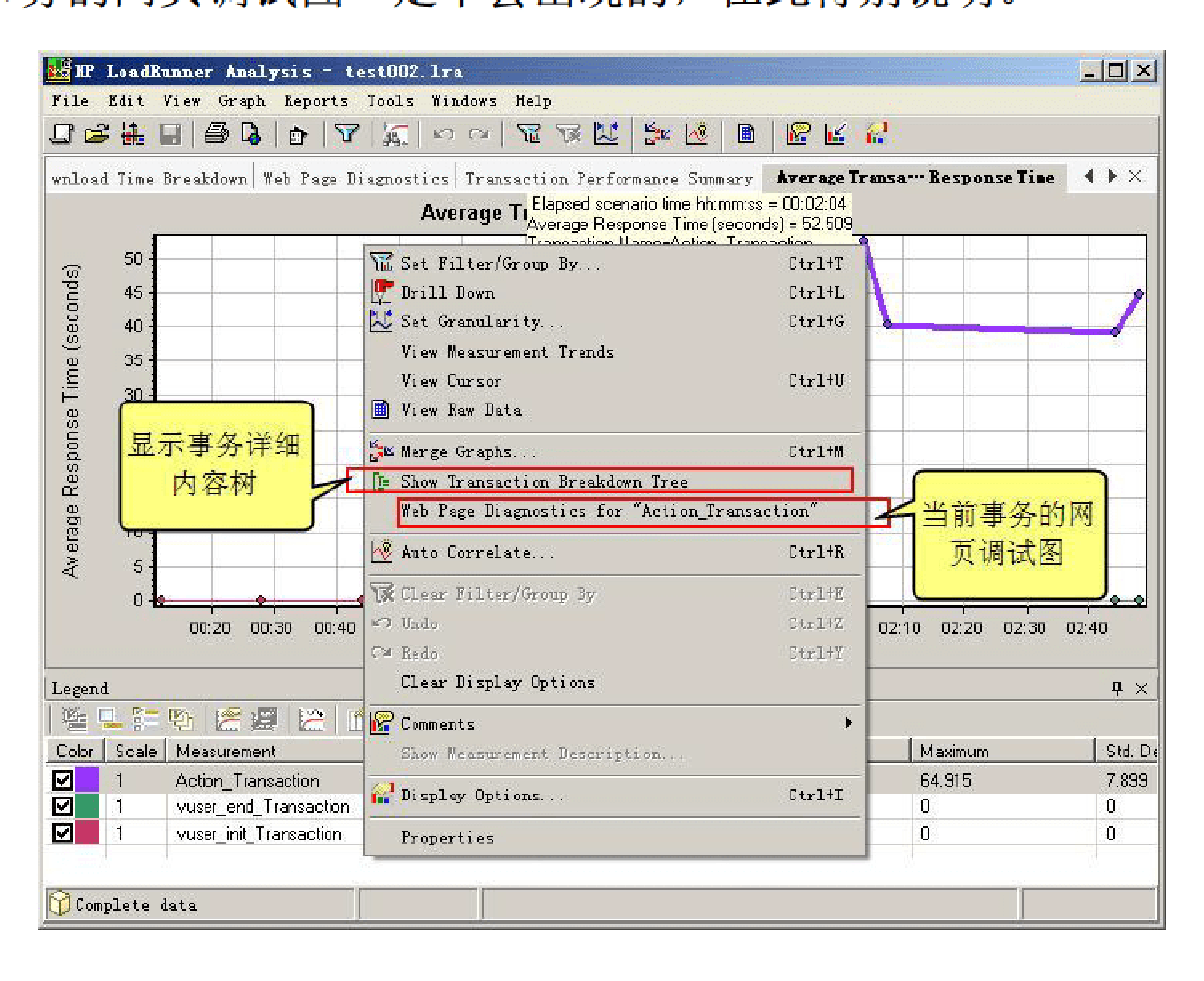


图 13-32　事务图的快捷菜单

单击图 13-32 中的显示事务详细内容树菜单项，分析器将在左边增加一个列表，所有

的事务度量都会列出，单击其中每一项，会在右边显示该度量单独的事务图表，如图 13-33 所示，该图是选择了初始化事务度量后的结果。由于图 13-32 包含了多条曲线，因此对于每一条曲线的变化规律很可能反映得不很细致，显示事务详细内容树可以方便地查看其中某条曲线。

【总图与详细图的对比】

在图 13-32 中，初始化事务度量的曲线（最左边处于图底部）几乎是一条直线，而图 13-33 中，曲线变化则陡峭得多，一目了然。因此，利用总图（图 13-32）和详细图（图 13-33）能全面地发现各事务度量的变化趋势。

如果需要返回总图，则可以在图 13-33 的图表空白区域右击，在弹出的快捷菜单中选择 Undo Filter/Group by（撤销过滤/分组）选项即可。如果要隐藏图 13-33 中左边的详细内容树（即度量列表），右击后在弹出的快捷菜单中选择 Hide Transaction breakdown tree（隐藏详细内容树）命令即可。

除了平均事务响应时间，在事务图类别中，还包含有其他的图表，下面将分别做讲解。对于这些图表，也都具有前文所讲述的快捷菜单项，可以在总图和详细图中切换。

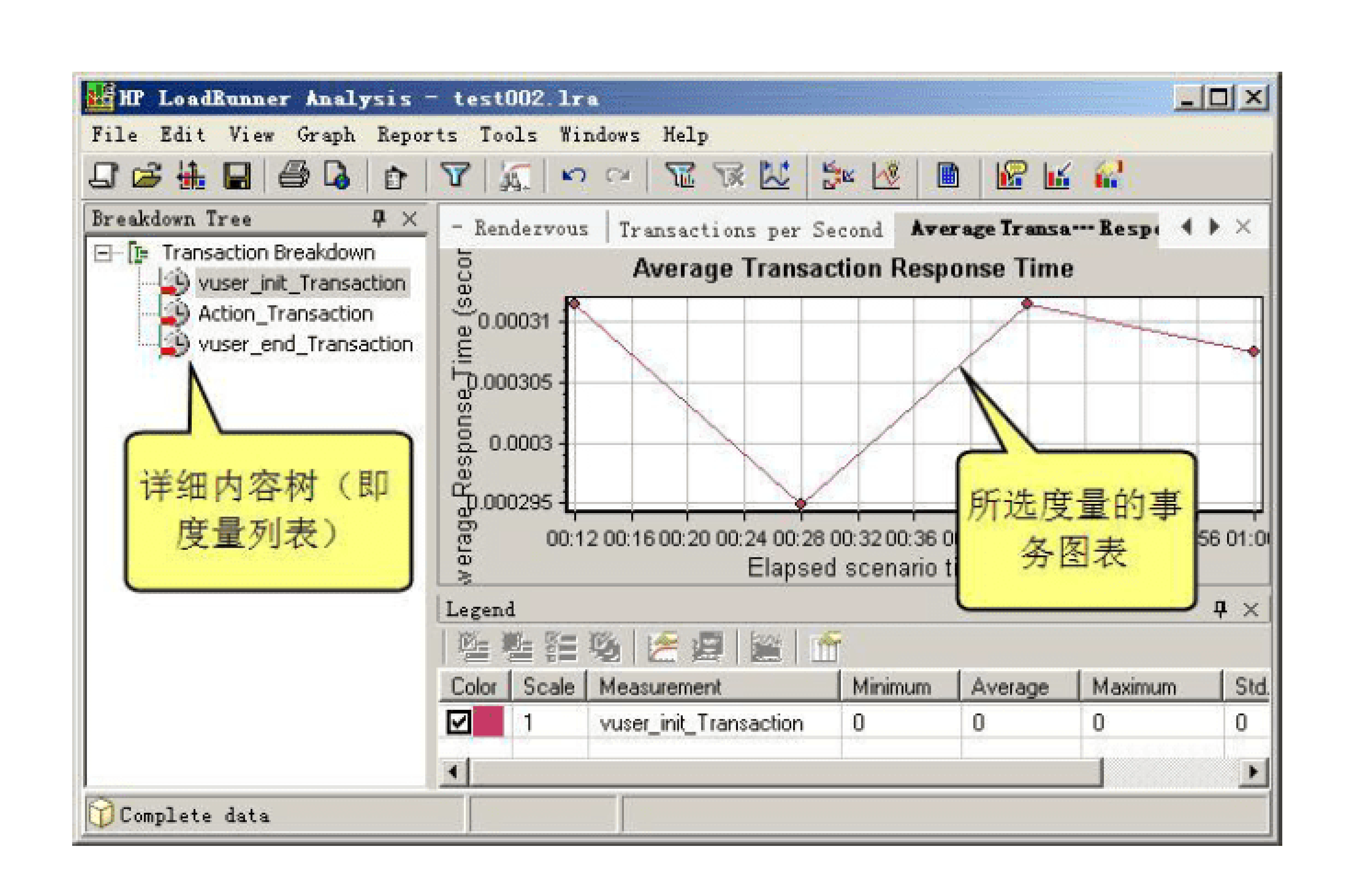


图 13-33　事务详细内容树可以对总图中的某条事务曲线进行显示

1. 每秒事务数量图

每秒事务数量图显示了某个或者多个度量在场景运行过程中，每秒发生事务数量的变化曲线，默认只显示通过的事务数量。该图能够让测试工程师了解场景中任意时刻的事务负载情况。如图 13-34 显示了 3 条曲线，分别代表了 Action_Transaction、vuser_end_Transaction 和 vuser_init_Transaction 的每秒事务数量图，X 轴为场景运行时间，Y 轴为每一时刻的事务数量。

2. 每秒事务总数量图

该图用来查看场景运行时间段内各时刻事务的总数量，只有一条曲线，其值为每秒事务数量图中各曲线数值的和。

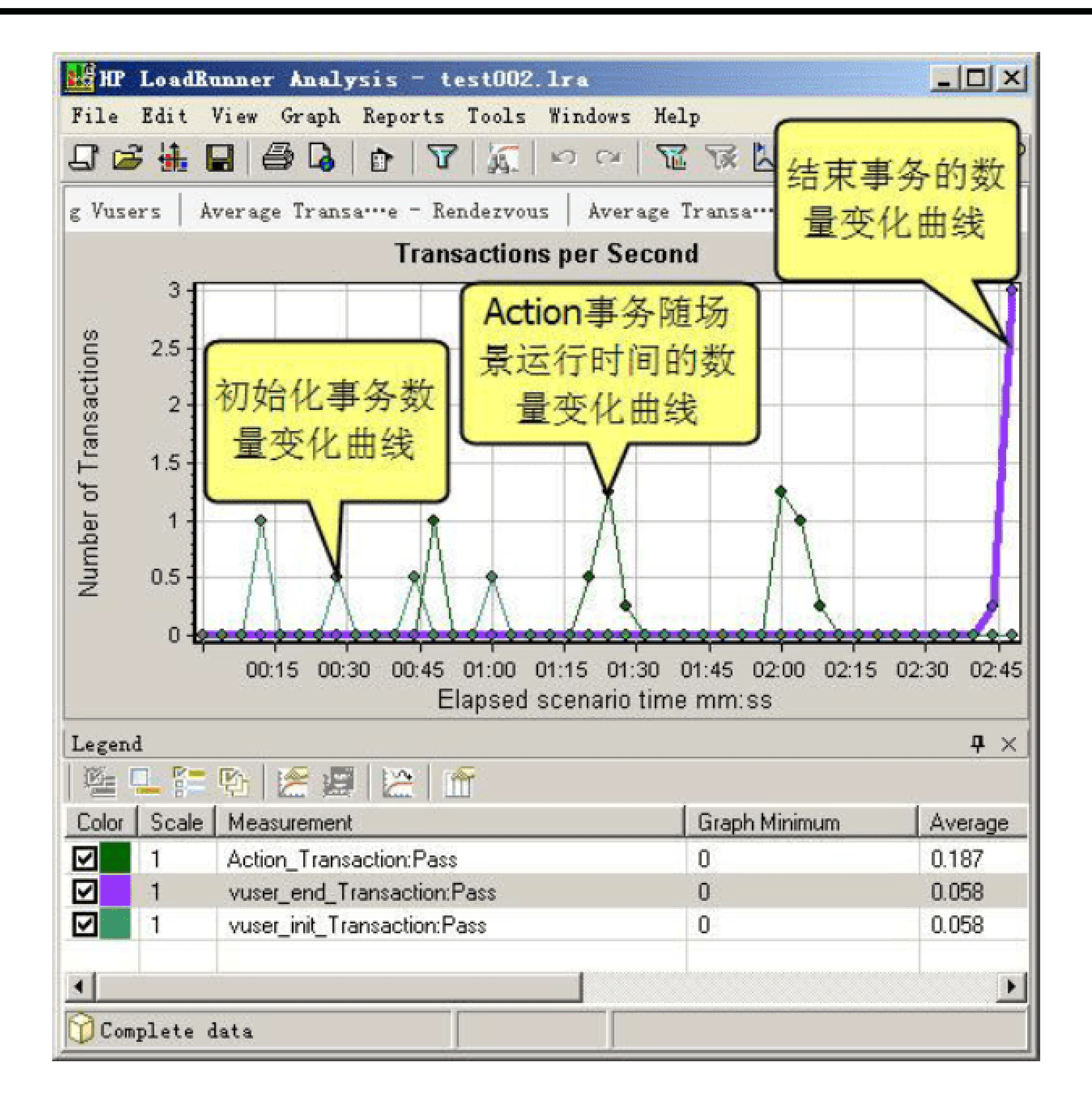


图 13-34　每秒事务数量图

3．事务概要图

该图用来查看场景运行中各度量的事务总数量，默认同样是只显示通过的事务，如图 13-35 所示。

4．负载下事务响应时间图

负载下事务响应时间图（如图 13-36 所示）将平均事务响应时间图和运行中虚拟用户数量结合起来，能显示当虚拟用户增加时，其所带来的工作负荷增加对于响应时间的影响。特别适合逐渐增加负荷的场景。

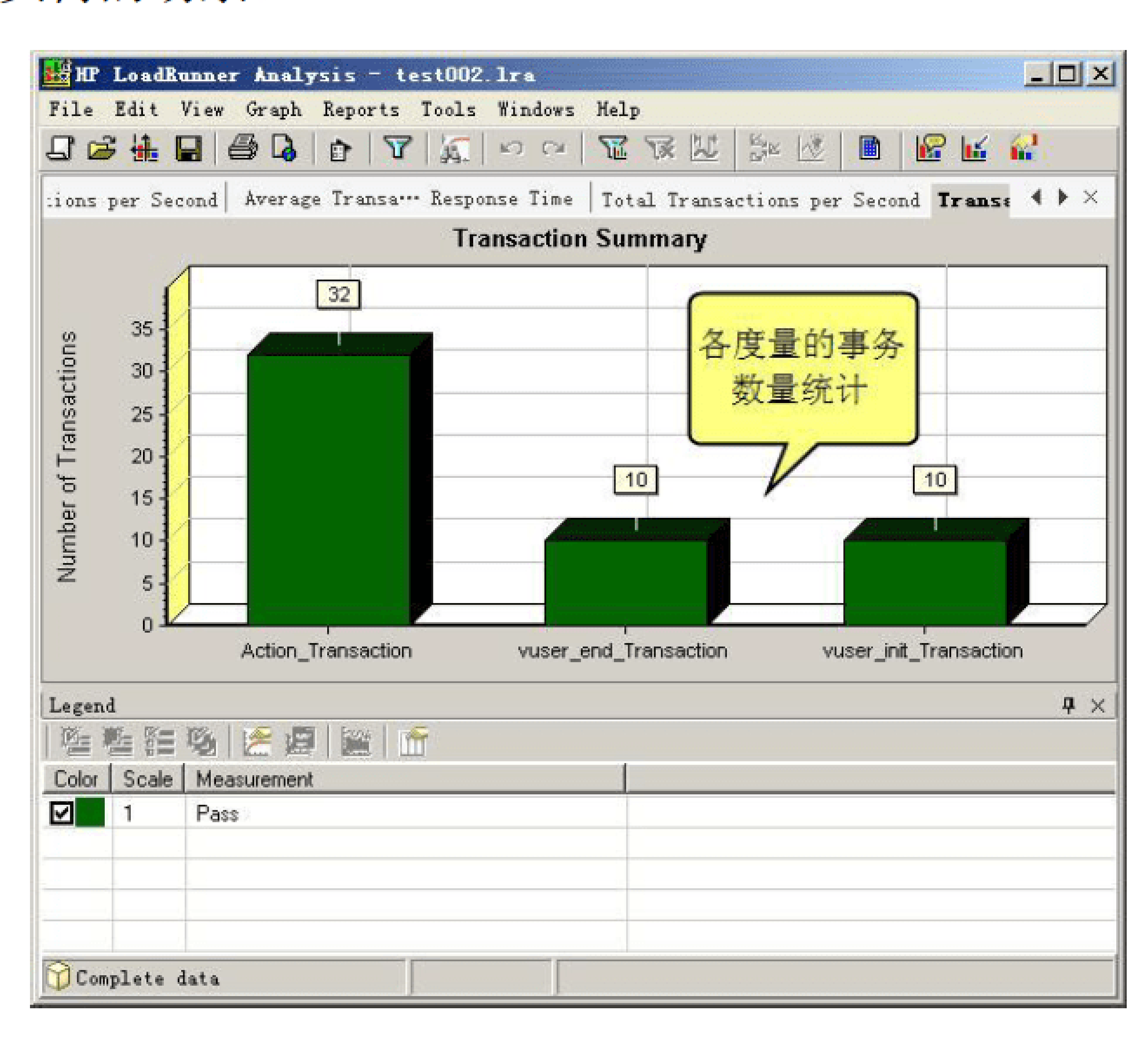


图 13-35　事务概要图

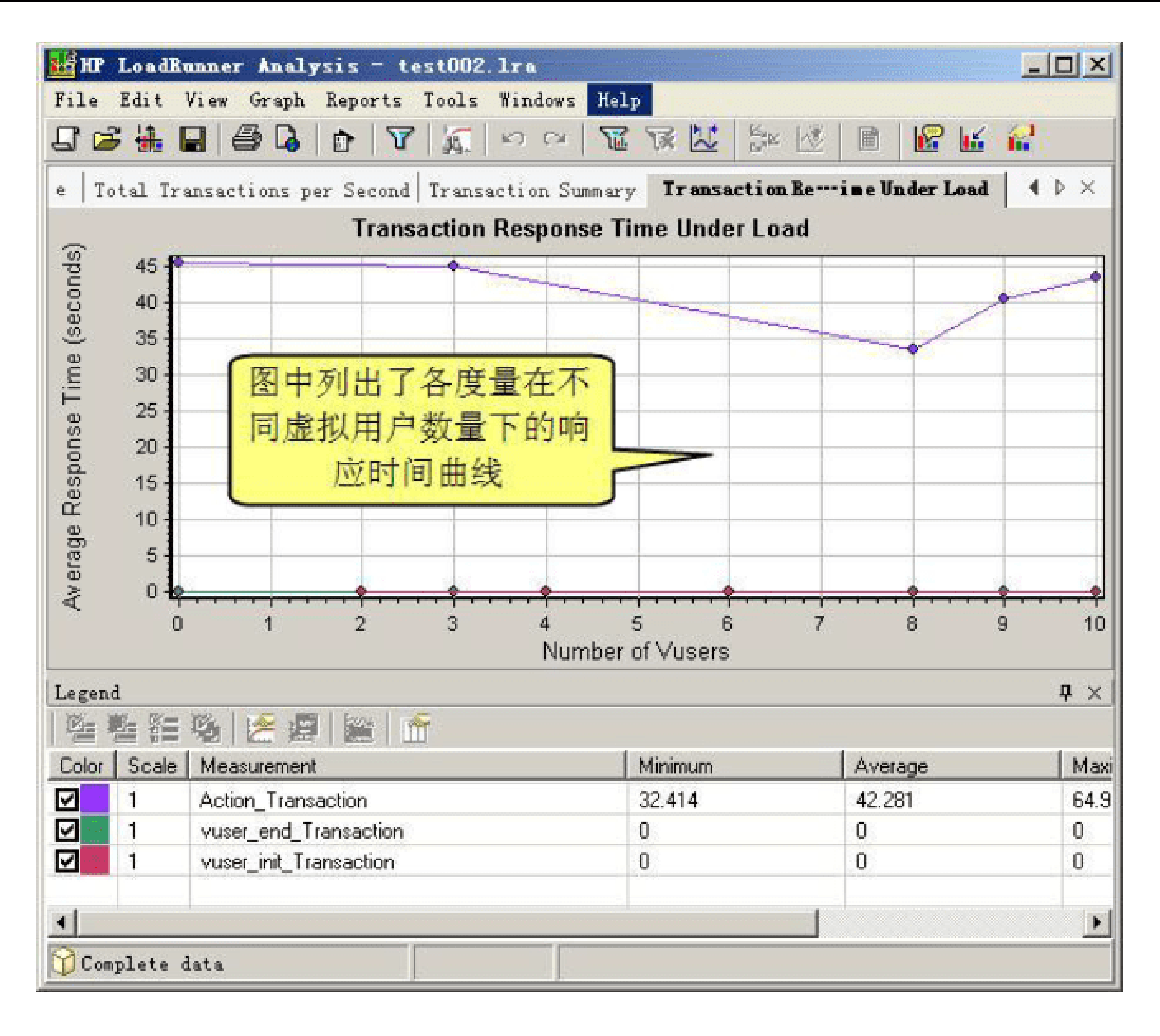


图 13-36　负载下事务响应时间图

5．事务响应时间百分比图

事务响应时间百分比图（如图 15-37 所示）用于查看某事务响应时间测试数值占整个数值范围的百分比情况，对判断响应时间是否符合标准具有一定的意义。事务响应时间百分比图中响应时间数值是递增排列的，因此可以判断小于某数值的比例。假如预先设置的事务响应时间合格标准是小于 50 秒（只是举例而言，实际的标准一般不会是这么大的数值），那么图 13-37 中将有 13%左右的数值不符合要求，如果这个比例可以接受，那么就可以判定这次场景运行结果达到了性能测试要求。

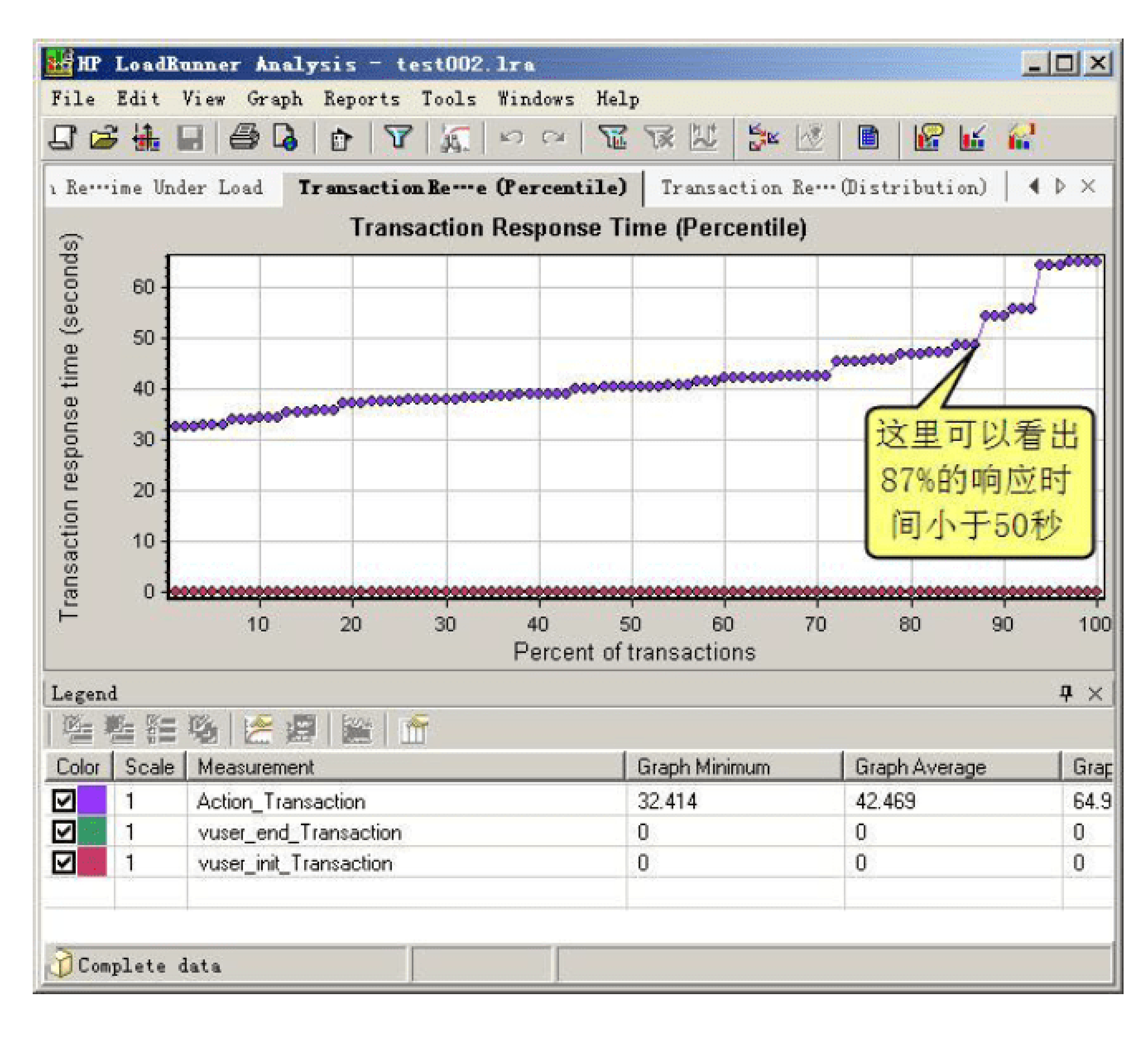


图 13-37　事务响应时间百分比图

事务响应时间百分比图的 X 轴表示递增的百分比数，直到 100%。Y 轴表示响应时间。在图表中的各数据点是按照递增的规律进行排列的，但是由于为了满足 X 轴每一个百分比均有响应时间数值对应的要求，Y 轴的数值不一定非常精确。

6．事务响应时间分布图

事务响应时间分布图（如图 13-38 所示）显示了响应时间数值的分布情况。X 轴代表最小单位为秒的响应时间，Y 轴表示事务数量。该图能够提示测试工程师大多数事务的相应时间在什么数值范围。

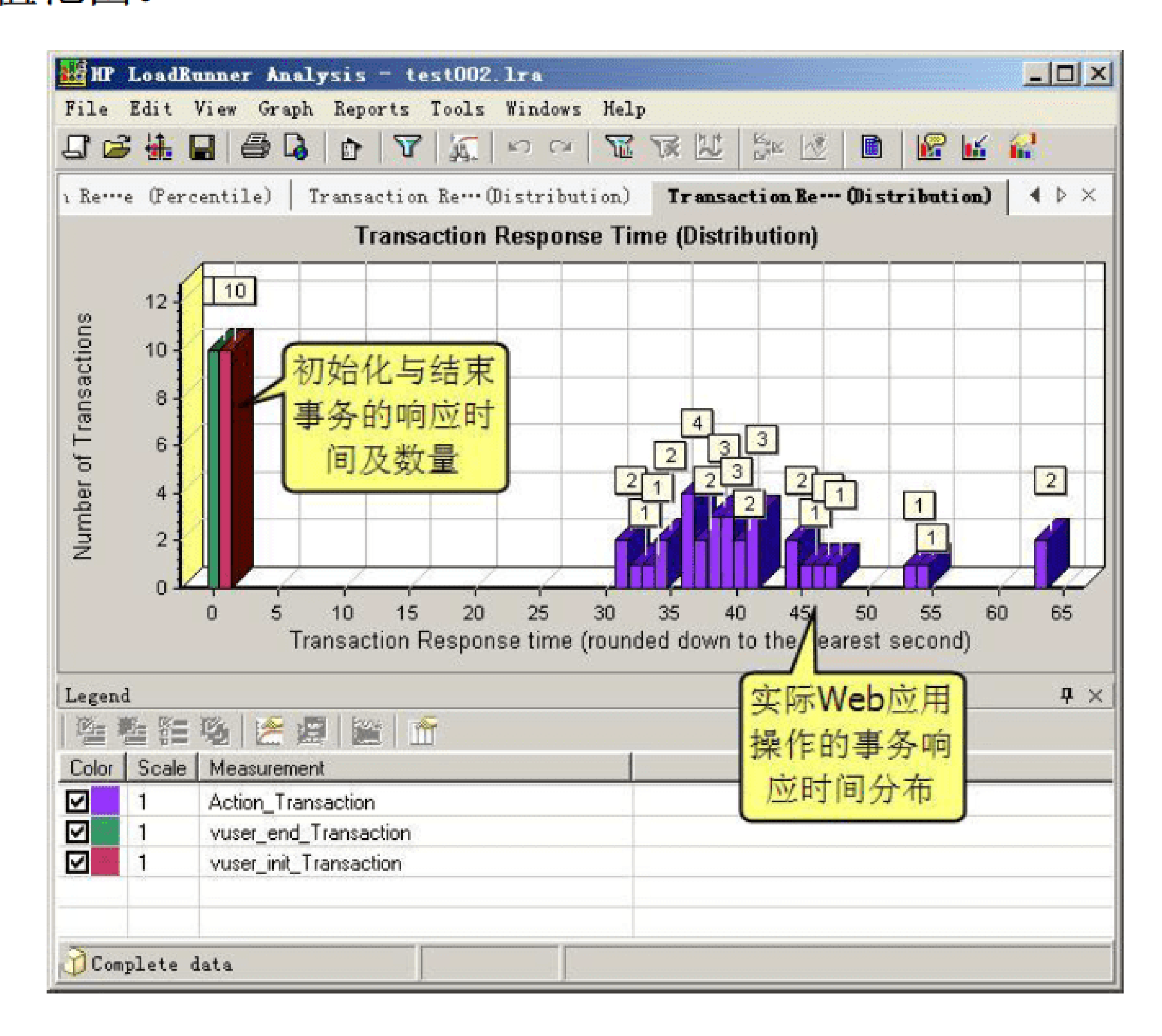


图 13-38　事务响应时间分布图

13.2.10　利用合并图进行图表的联合分析

合并图可以用来比较同一场景内不同图表，以发现各图数据之间的关系。本节以介绍过的虚拟用户图和平均事务响应时间图为例，介绍合并图的技巧。

通过合并图，可以将所有具备相同 X 轴的图表数据并列于一个图表中，便于发现各自曲线的共同趋势、相反趋势等，从而进一步发现性能变化的规律。一般来说，各个图表的 X 轴度量均为场景运行时间，因此合并图技巧应用是很广泛的。

在图 13-32（事务图）中右击，在弹出的快捷菜单中选择 Merge Graphs（合并图表）选项，即可打开合并图的设置窗体；同时在分析器顶部的工具栏中，也有合并图的图标，如图 13-39 所示。单击合并图图标按钮后，弹出合并图设置窗体，如图 13-40 所示。

如果在图 13-40 中没有需要被合并的图表，则需要考虑如下两种情况并加以解决：

- 需要被合并的图表没有添加到当前分析会话中。按照本章前文所述，添加该图表即可。
- 需要被合并的图表与当前图表没有可共享的 X 轴度量。这种情况下是无法合并图表的，只能利用交叉结果图等其他方法来发现性能变化规律。交叉结果图将在

13.2.10 节介绍。

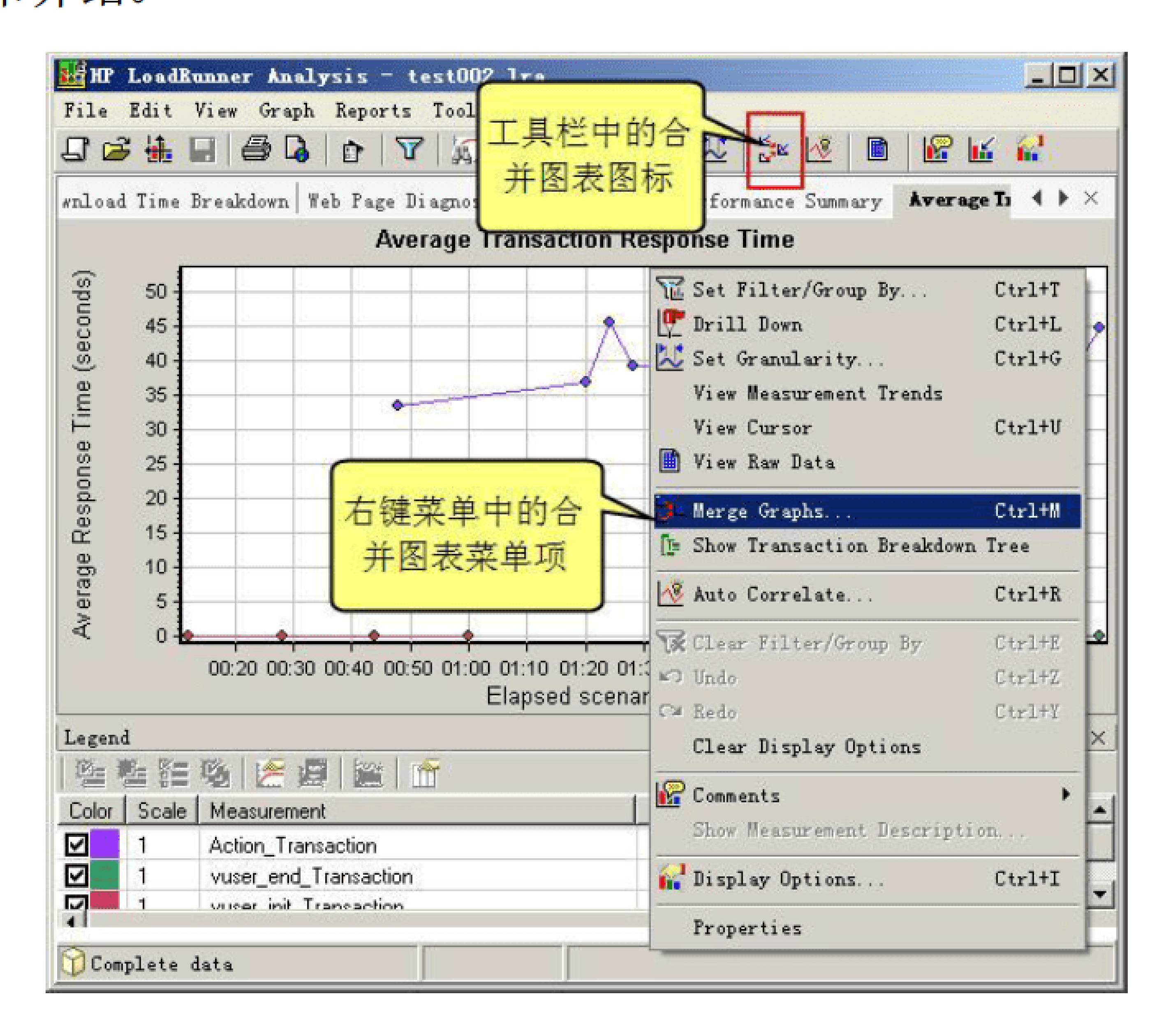


图 13-39　工具栏与右键菜单中的合并图表功能

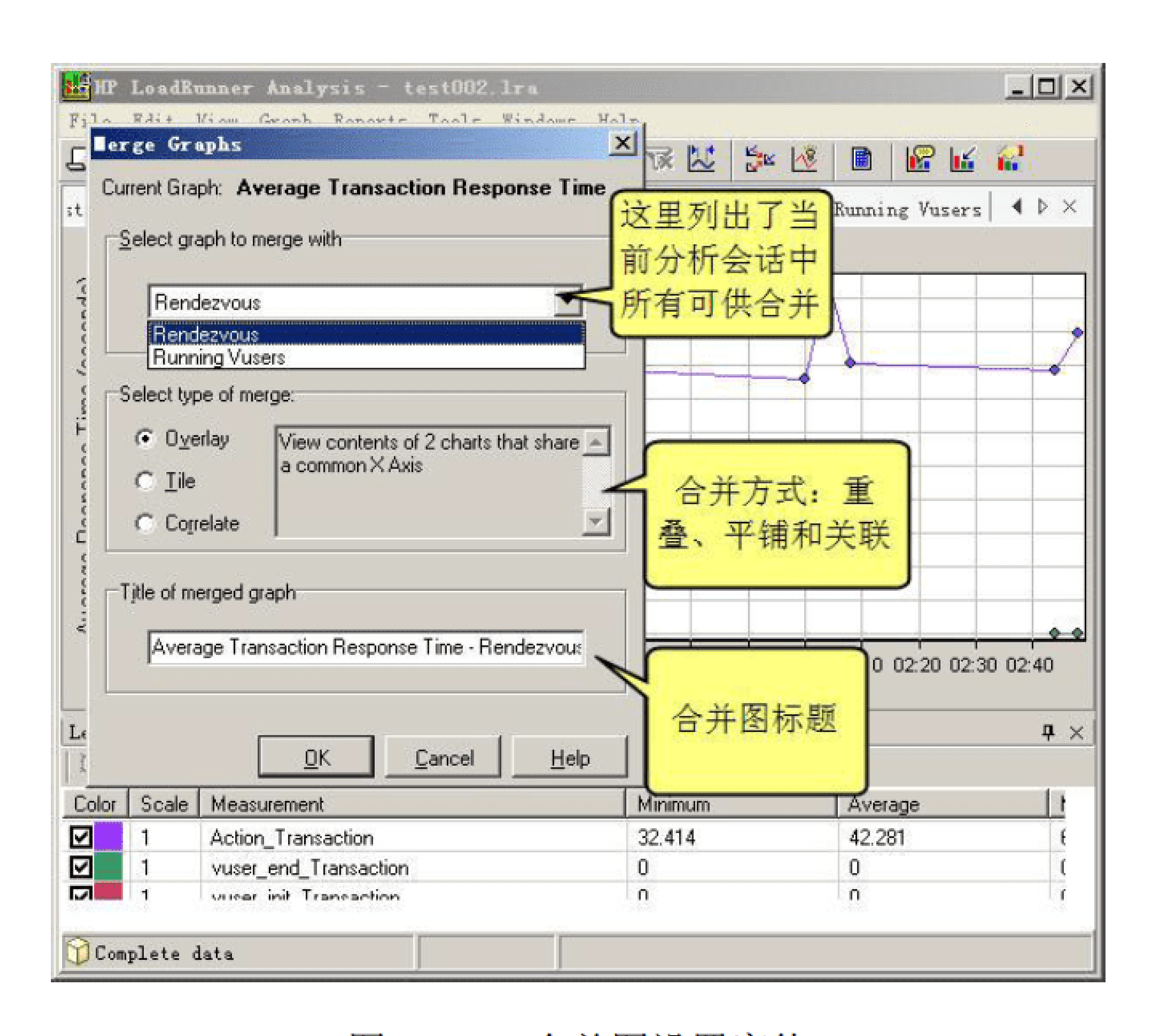


图 13-40　合并图设置窗体

在图 13-40 中我们选择运行中虚拟用户图为例，和当前的平均事务响应时间图表进行合并，新图表的标题保持默认设置即可，它实际就是将两个图表的标题联合了起来。单击图 13-40 中的 OK 按钮后合并图表成功，新图表如图 13-41 所示。

分析器提供了 3 种图表合并显示方法：

- 叠加法（Overlay）：被合并的两个图表 X 轴共享，各自的 Y 轴标尺处于新图表的左边与右边边界。图 13-41 就是采用 LoadRunner 默认的叠加法合并而成的。
- 平铺法（Tile）：被合并的两个图表进行平铺布局，共用 X 轴数据，但一个处于另

一个之上，如图 13-42 是图 13-41 中两条曲线采用平铺法进行合并的结果。

图 13-41　合并后的新图表（采用默认叠加法）

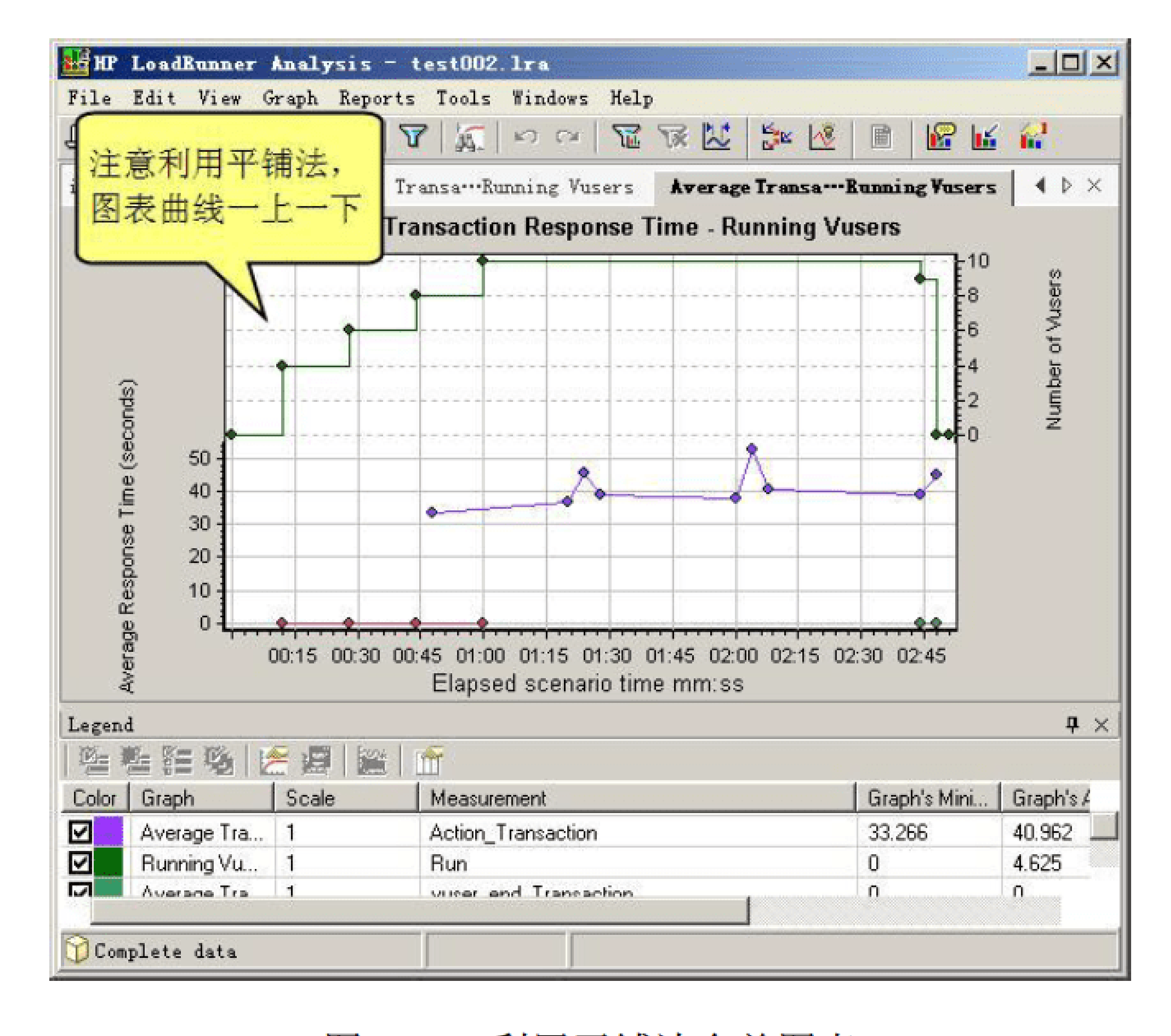


图 13-42 利用平铺法合并图表

- 关联法（Correlate）：根据两个图表的 Y 轴绘制合并后的新图表，当前图的 Y 轴变为合并后新图表的 X 轴，而被合并图的 Y 轴则作为合并后新图表的 Y 轴，如图 13-43 所示。

【关联法对于被合并图的限制】

请注意对于关联法，如果被合并的图中包含多条曲线，类似这里平均事务响应时间图表中的情形（3 个事务，3 条事务曲线），LoadRunner 将提示只能选择一条曲线进行关联，因此有必要通过过滤/分组功能（Set Filter/Group by）令图表只包含一条曲线。

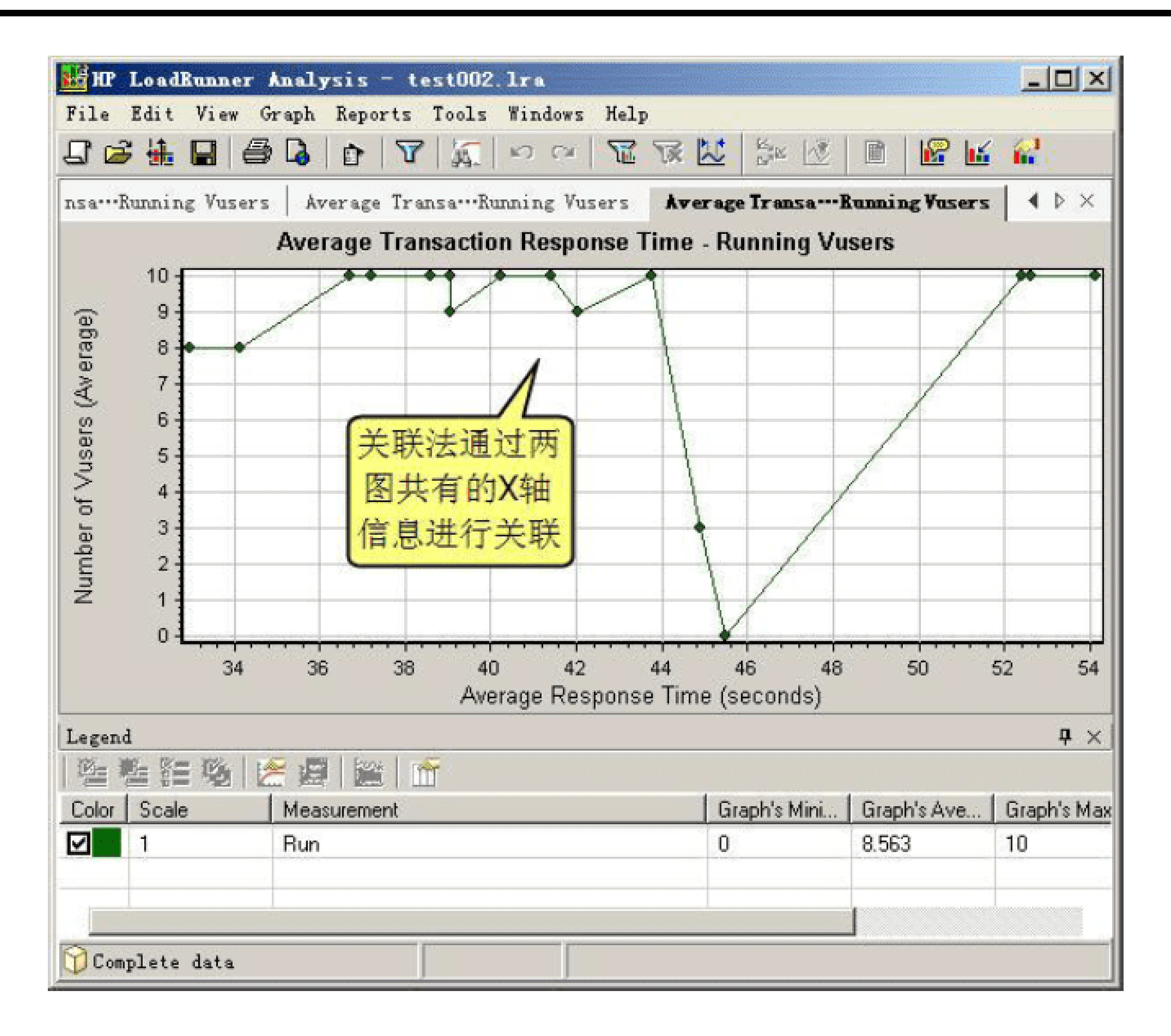


图 13-43　利用关联法合并图表

在关联法所生成的图 13-43 中，由于去除了场景时间的变化因素，因此实际上考察的是平均响应时间测试结果范围内，虚拟用户各数量的分布情况。图 13-43 出现了一个较大的 V 型下沉峰谷，是因为当前场景结束时平均虚拟用户数量急剧下降造成的。通过图 13-43，可以判断出虚拟用户处于某数值水平时，平均响应时间至少会大于的数值。

【多重合并】

合并后的新图表还可以继续合并其他的图表，但是，这种情况只对采取默认的叠加法进行初次合并的新图表有效。

将平均事务响应时间与虚拟用户图表进行合并，可以发现虚拟用户增加对于事务响应时间的影响规律。

13.2.11　利用交叉结果图进行多场景的横向分析

在开发人员对代码采取某些优化之后，性能测试工程师需要用数据来证明优化的成果。在这样的情况下，需要对多场景或者多次分析会话的结果进行比较。LoadRunner 提供了交叉结果图这一工具来实现此功能。在 LoadRunner 分析器中选择 File 选项，如图 13-44 所示。

选择其中的 Cross with Results（交叉结果）选项，弹出设置对话框如图 13-45 所示。在 Result list（结果列表）对话框中列出了当前分析会话中的现有结果文件（lrr 文件）。

单击 Add 按钮，将弹出查找文件对话框，找到另外的某个结果文件。单击 OK 按钮后即可以将其添加到列表当中。默认情况下，图 13-45 中创建新的分析会话是选中的，如果去除勾选，则会将生成的交叉结果图保存到当前的分析会话之中。

将多个结果文件进行交叉后，将新产生多个图表，其中每个图表都将显示各结果文件中的数据以供对比。如图 13-46 显示了某两个场景的运行结果进行交叉后新生成的虚拟用户图。对于这样的图表，同样可以使用其右键菜单的过滤/分组、设置粒度、下钻等前面讲

过的功能。

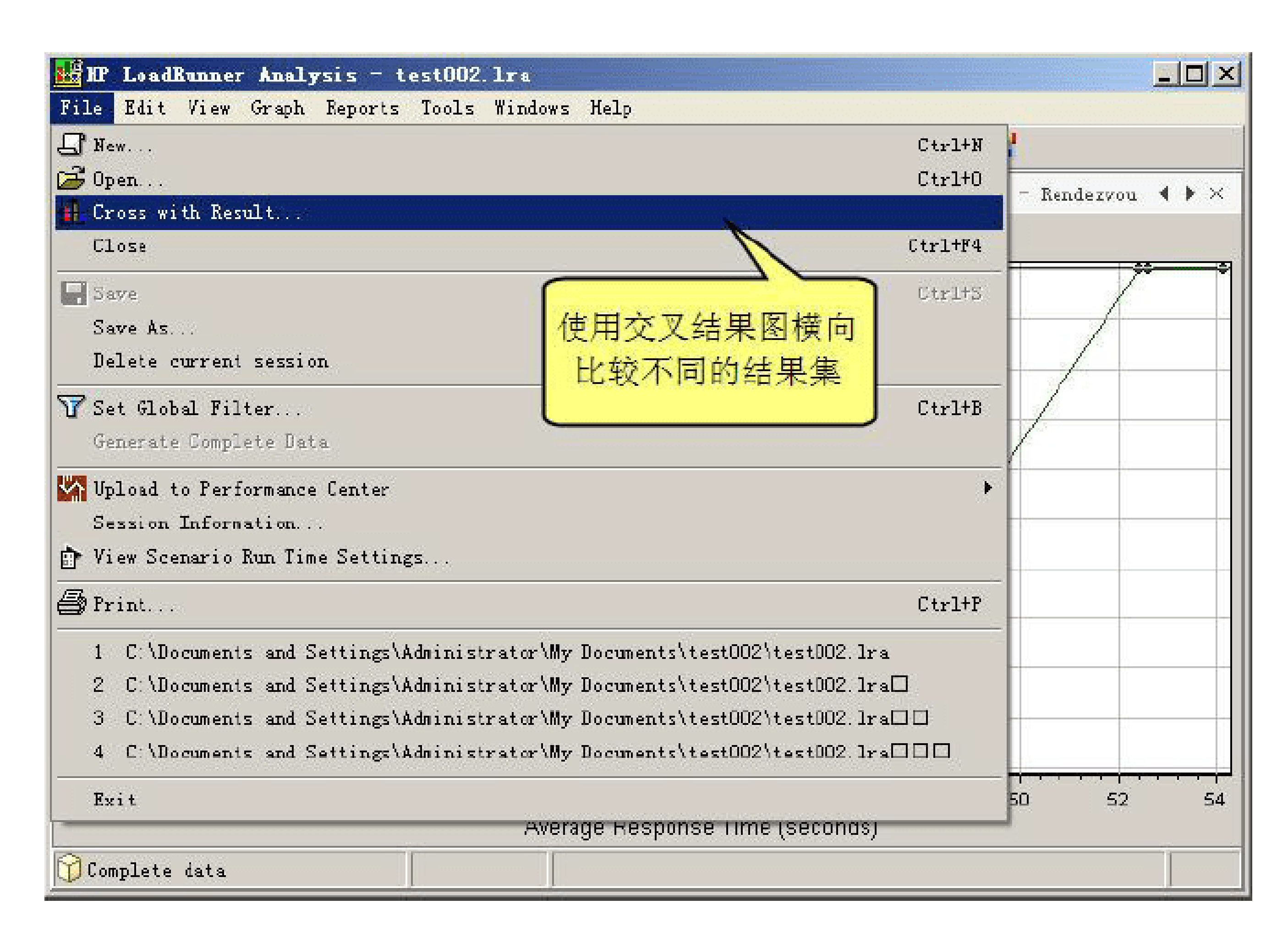


图 13-44　分析器中的交叉结果图菜单项

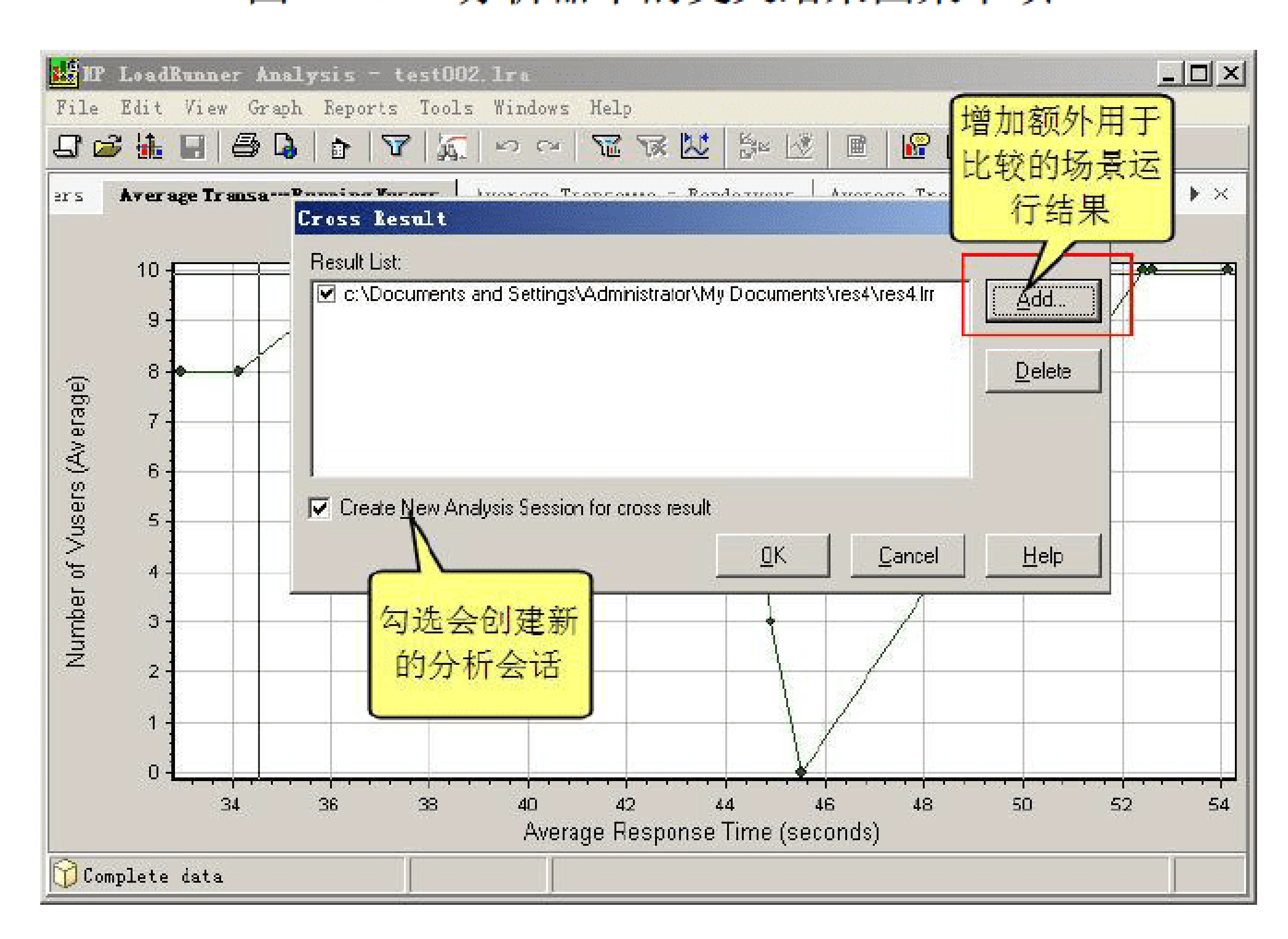


图 13-45　添加新结果文件以便交叉分析

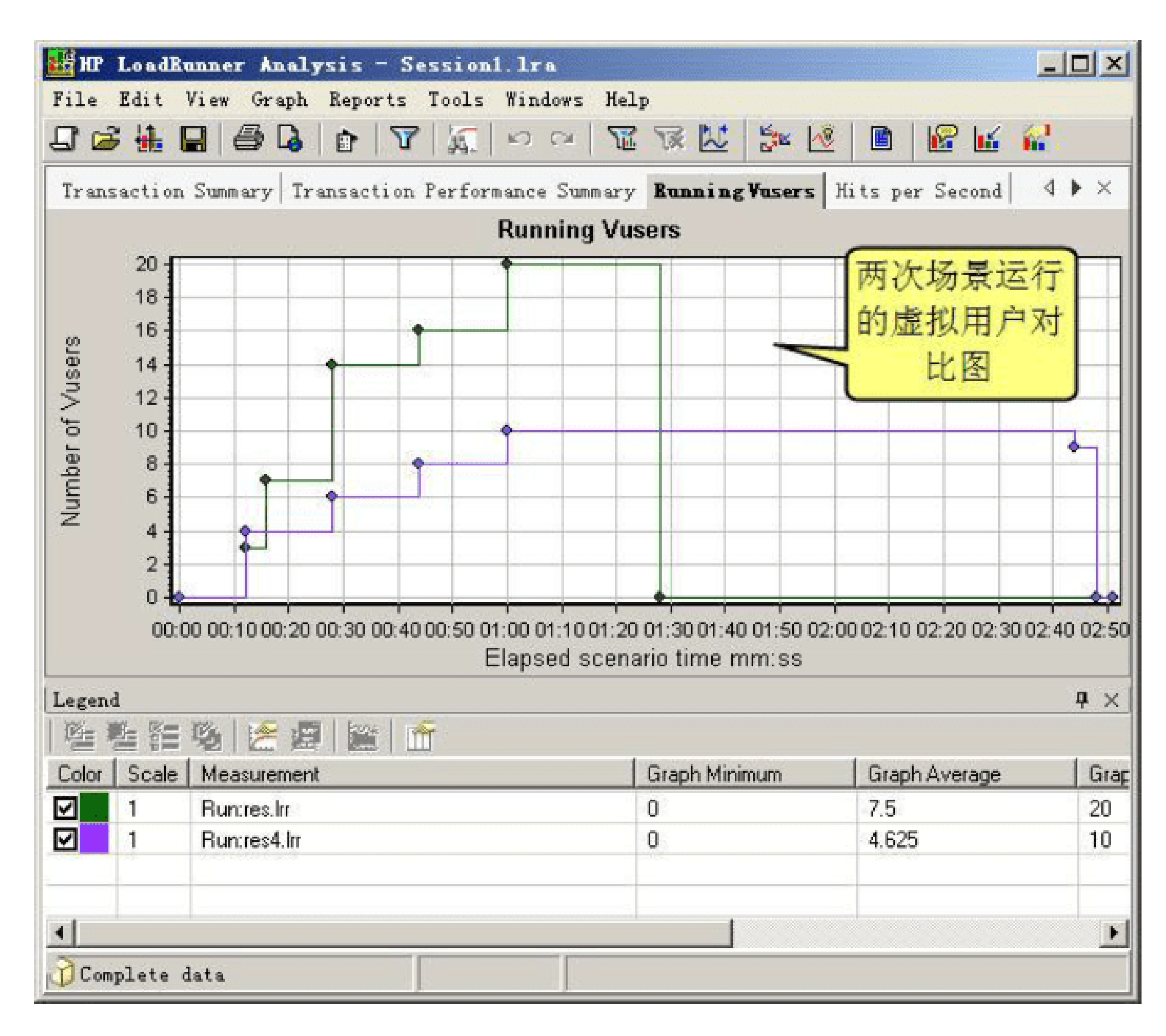


图 13-46　两次场景运行结果的虚拟用户对比图

【交叉结果图的作用】

交叉结果图总体说来有 3 个作用：对在不同的硬件上，运行同一场景所产生的多个结果文件进行交叉，以评估硬件性能；对在相同硬件上，部署不同版本的 Web 应用后运行同一场景所产生的多个结果进行交叉，以评估不同版本的优化效果；对在相同硬件上，部署相同版本的 Web 应用后，运行同一场景但参数不同（比如虚拟用户不同、持续时间不同等等）的多个结果进行交叉，以评估 Web 应用的性能极限。

13.2.12　网络资源图（Web Resources 图）

在介绍完事务图和若干图表分析技巧之后，本节将继续讲解 Web 应用相关图表的另一种类型：网络资源图。

网络资源图可以提供 Web 服务器的性能信息，它包含如下等多种图表：

- ❑ 每秒点击次数（Hits per second）。
- ❑ 吞吐量（Throughput）。
- ❑ HTTP 状态码概要（HTTP Status Code Summary）。
- ❑ 每秒 HTTP 反馈（HTTP Responses per second）。
- ❑ 每秒页面下载（Page Downloaded Per Second）。
- ❑ 连接数（Connections）。
- ❑ 每秒连接数量（Connections Per Second）。

这里将重点介绍最为重要的吞吐量和每秒点击次数图。HTTP 状态码概要、每秒 HTTP 反馈等在第 12 章的测试概要中已经介绍过，而其余图表均较好理解，在实际工作中很快就可以熟悉。

1．吞吐量图表

吞吐量显示了场景运行当中每一秒内服务器上的吞吐量，单位为字节数。该图的 X 轴表示场景运行经过时间，Y 轴则是虚拟用户从服务器上得到的字节数，即吞吐量数值，如图 13-47 所示。

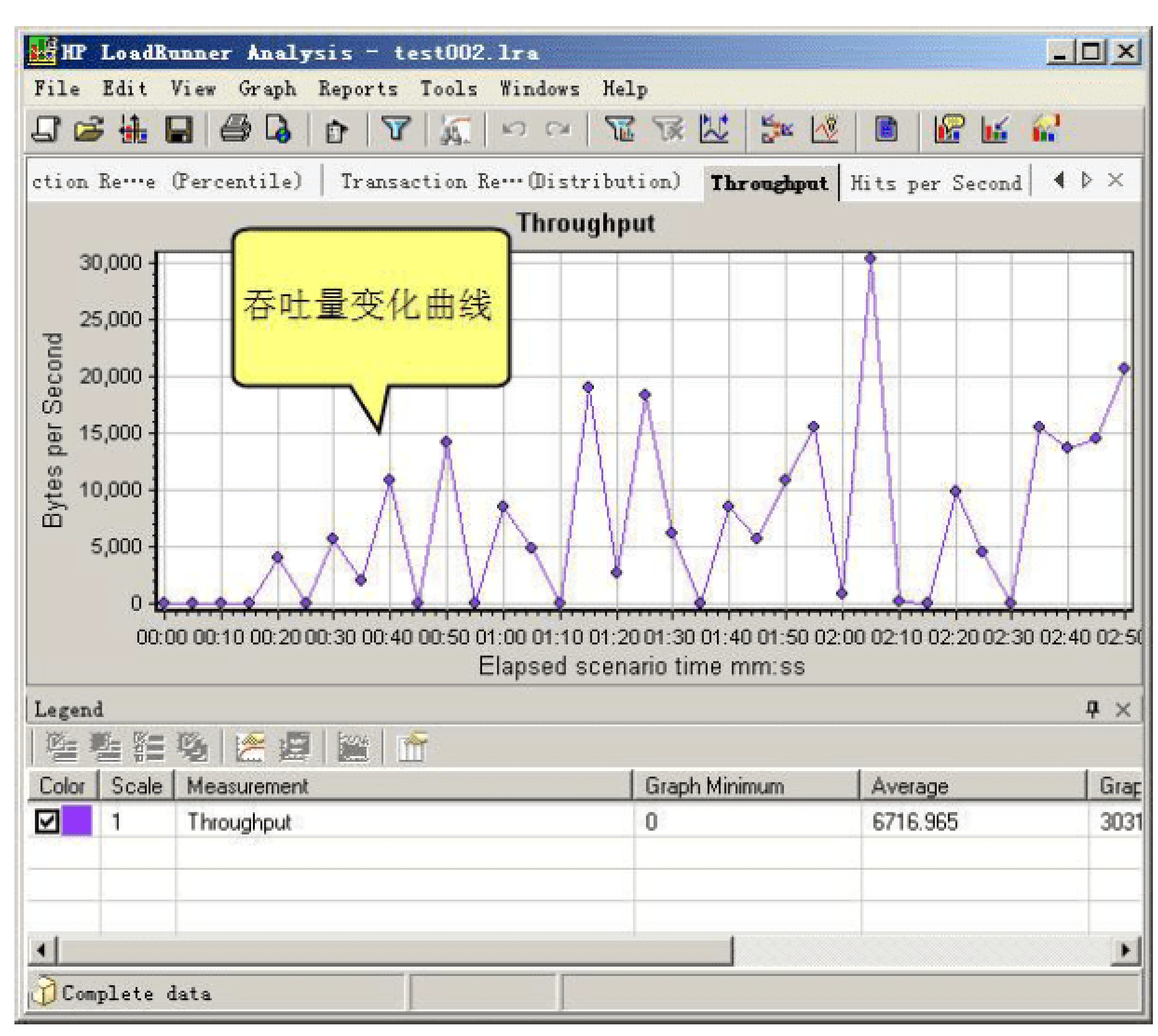


图 13-47　吞吐量图

可以将吞吐量图与平均事务响应时间图进行比较，以获得吞吐量对于事务性能的影响。

2．每秒点击次数图

每秒点击次数图显示了场景运行当中每一秒内，虚拟用户向 Web 服务器提交的 HTTP 请求数。该图使用点击次数来评估虚拟用户产生的工作负荷，如图 13-48 所示。

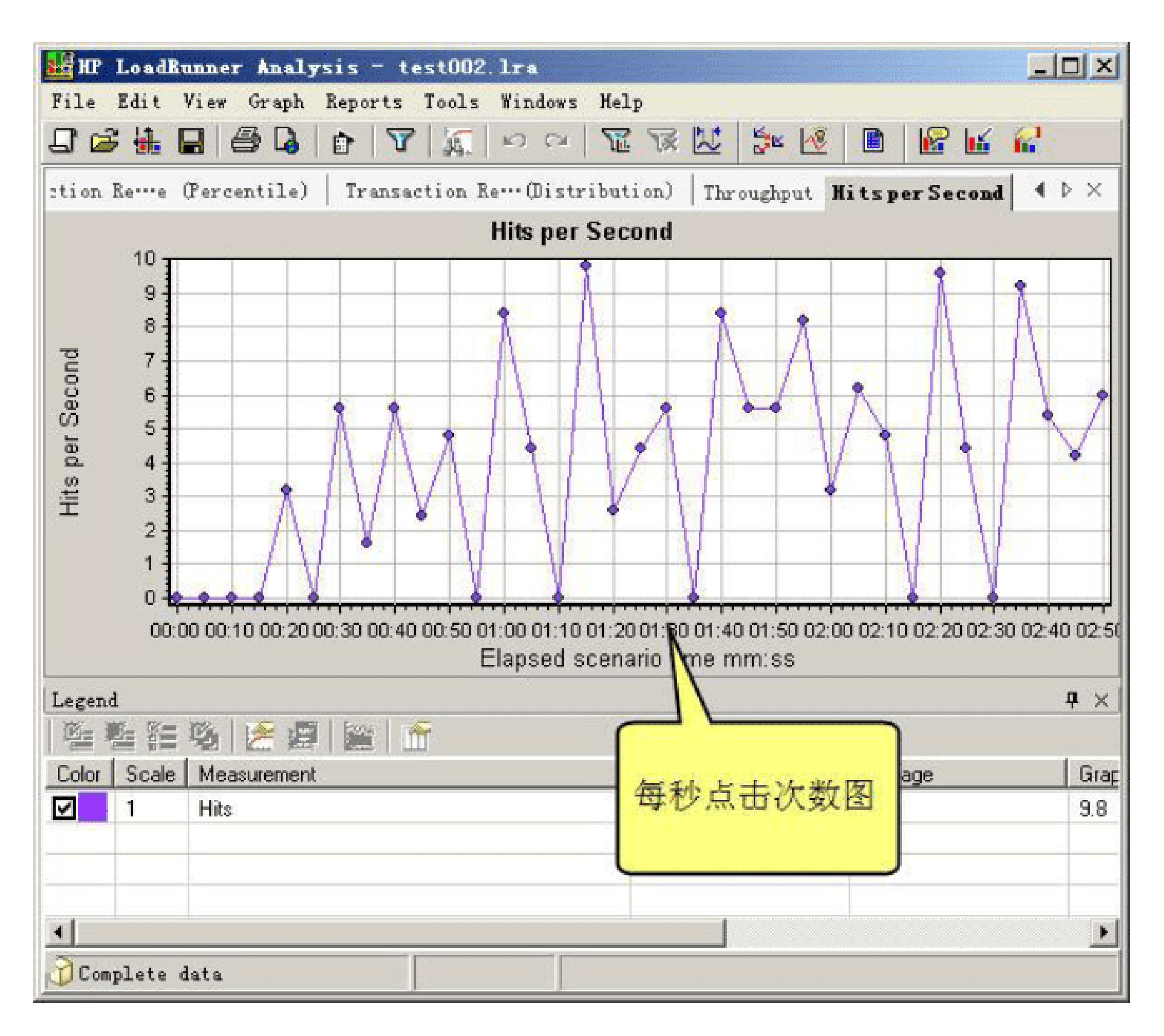


图 13-48　每秒点击次数图

通过合并图，可以将每秒点击次数图与平均事务响应时间图进行比较，以获得点击次数的多寡对于事务性能所产生的影响，如图 13-49 所示。

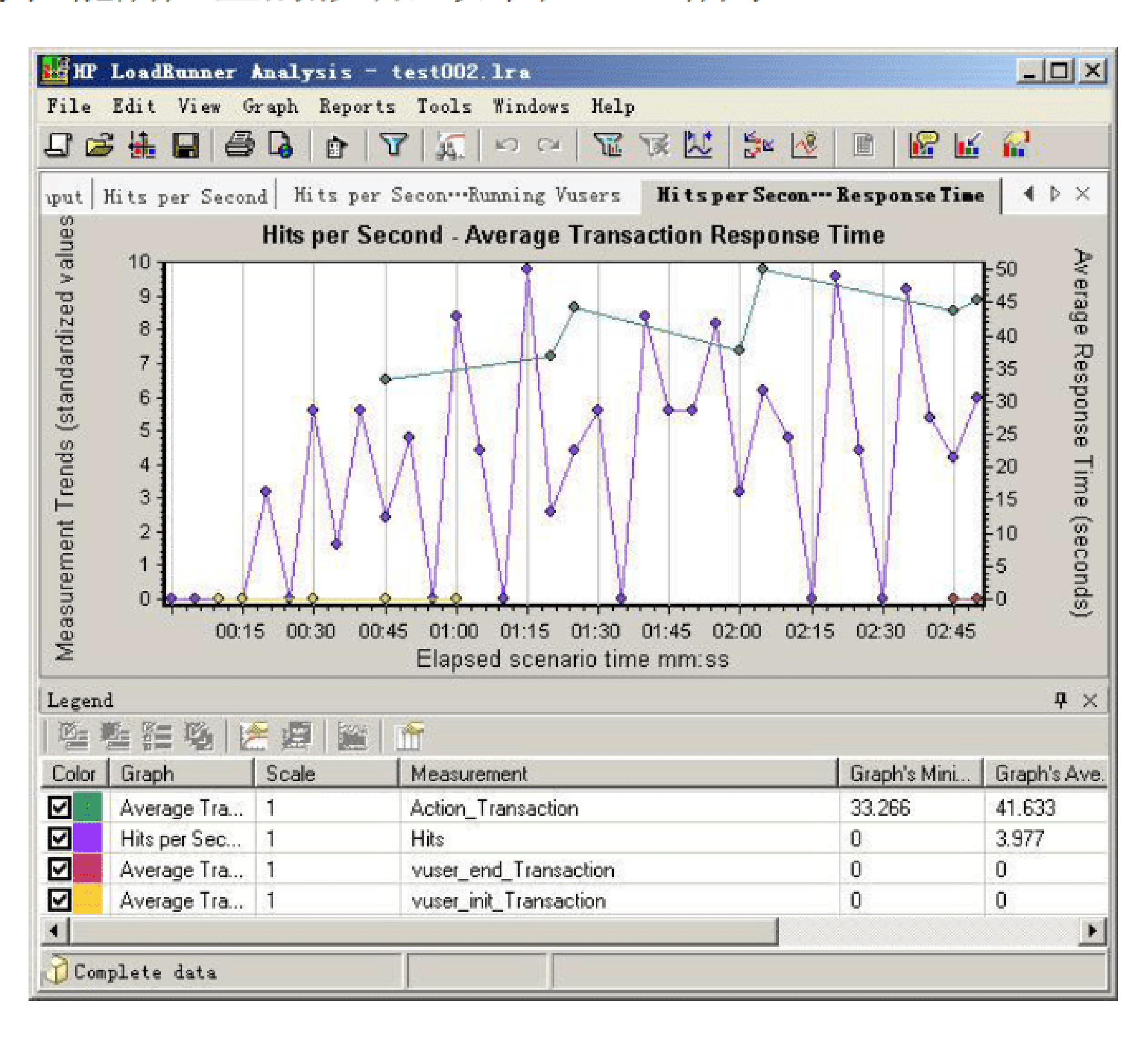


图 13-49　每秒点击次数与平均事务响应时间的合并图

从图 13-49 中可以发现，每秒点击次数的增长与对 Web 应用进行操作的事务（图中上部的短折线）同时发生。

【查看度量趋势】

由于图 13-49 中的多数折线都比较陡峭，因此最左下方的线由于 Y 轴坐标的关系近似为一条直线。我们可以在图表空白处右击，在弹出的快捷菜单中选择 View Measurement Trends（查看度量趋势）选项的方法来显示它的变化规律，如图 13-50 所示。但是，查看度量趋势只能适用于线图，前面内容中用户概要那样的饼图是无法应用的。

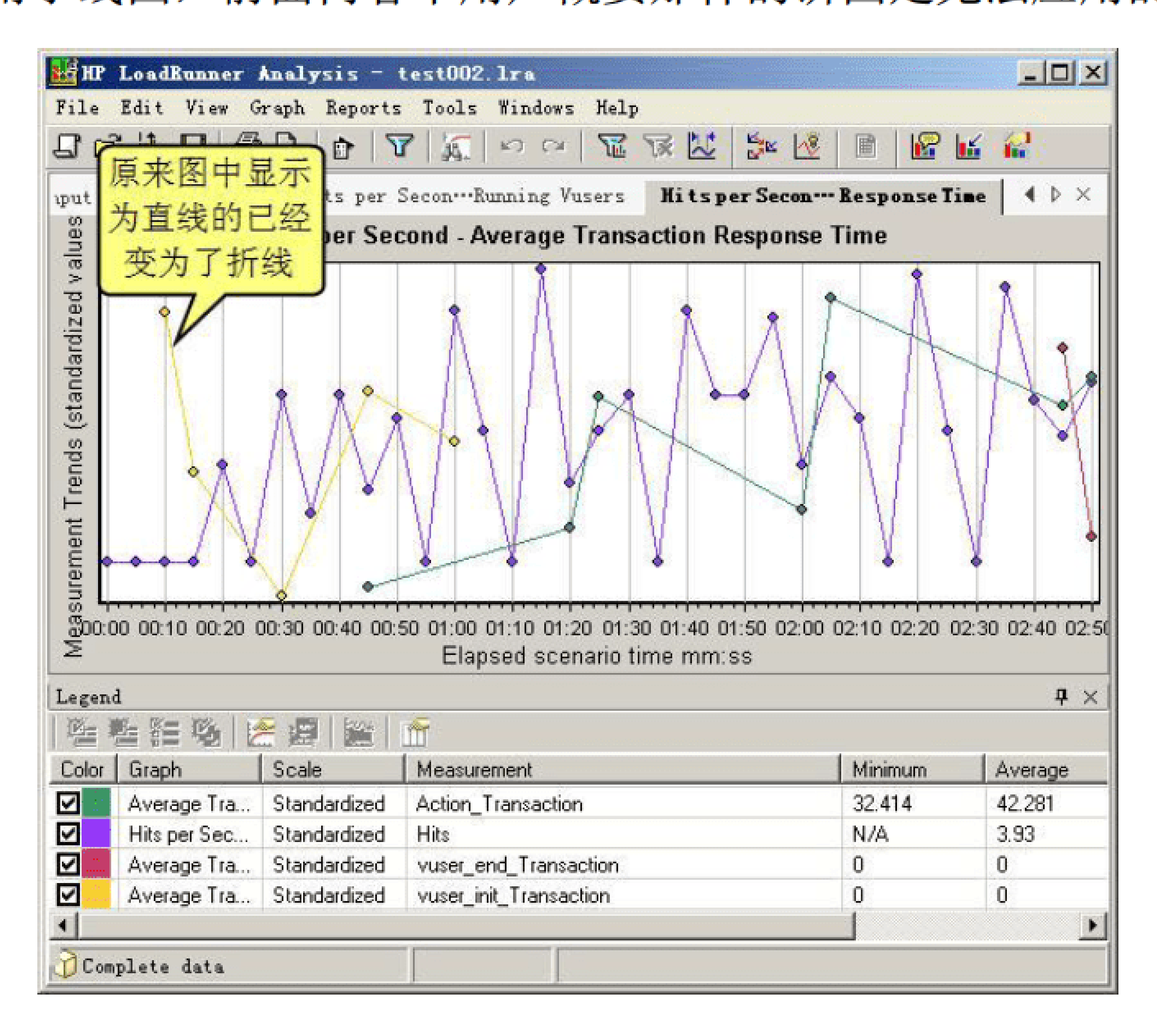


图 13-50　查看度量趋势

13.2.13　网页调试图（Web Page Diagnostic 图）

网页调试图用来分析 Web 应用中页面的具体内容是否对事务响应时间有所影响，它能够提供 Web 应用中各个被监控网页的性能信息。

要在分析器中能够使用网页调试图，则必须在控制器场景配置中将网页调试启用。打开 LoadRunner 控制器（Controller），打开待运行的场景，选择 Diagnostic|Configuration（调试|配置）命令，弹出对话框如图 13-51 所示，单击 Web Diagnostic（网页调试）旁的 Disable（启用）按钮即可。

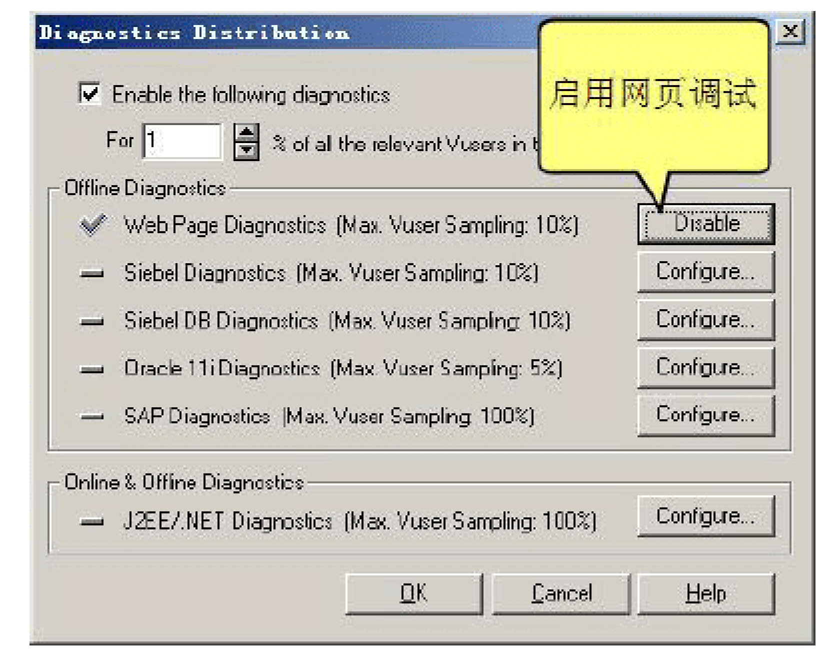


图 13-51　在控制器中启用网页调试

当在控制器中为场景启用网页调试之后，再运行场景，就可以得到网页调试图的结果了。

网页调试图包括如下多个图表：

- ❑ 网页调试图（Web Page Diagnostic）。
- ❑ 页面组件细分图（Page Component Breakdown）。
- ❑ 页面组件随时间变化细分图（Page Component Breakdown（over time））。

- 页面下载时间细分图（Page Download Breakdown）。
- 页面下载时间随时间变化细分图（Page Download Breakdown（over time））。
- 第一次缓冲细分时间图（Time to First Buffer Breakdown）。
- 第一次缓冲随时间变化细分图（Time to First Buffer Breakdown（over time））。
- 已下载组件尺寸图（Downloaded Component Size（KB））。

本节将重点介绍网页调试图、页面组件细分图和页面下载时间细分图。

1. 页面组件细分图

页面组件细分图描述了场景运行所需要操作的各页面（即组件），以及页面内各元素的平均下载时间（以秒为单位），如图 13-52 所示。这是网页调试图中最基本的图表，可以使得测试工程师对于场景运行、事务处理需要调用的页面有更清楚的认识。

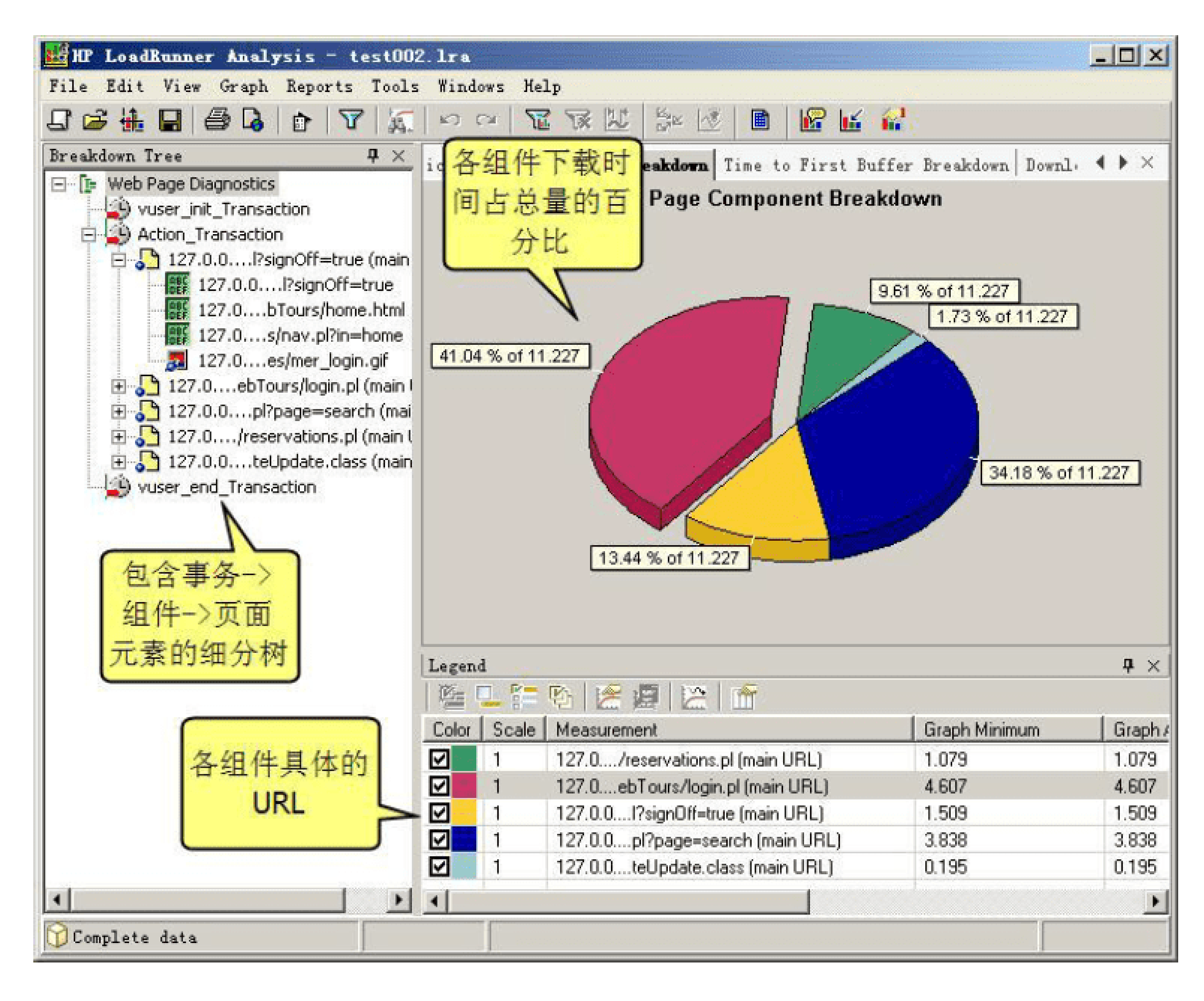


图 13-52　页面组件细分图

从图 13-52 中可以看到，各事务运行所调用的 URL（即组件）被列在了左侧的细分树和下方的图例当中。每个组件所包含的图片、访问链接等页面元素在细分树视图中还可以通过单击组件名称展开。界面右边的饼图显示了各组件下载时间占总时间的百分比，从中可以发现哪个组件耗费了更多的时间。在图 13-52 中，登录和搜索组件分别占总下载时间的 41%和 34%，因此这两个组件相对而言更有可能存在性能问题。

在图例视图中，可以对各个组件的下载时间统计值（最大值、最小值等）进行排序，从而更容易发现问题。

2. 页面下载时间细分图

页面下载时间细分图以柱图的形式显示了每个页面组件下载时间的各个子步骤细分数值，如图 13-53 所示。

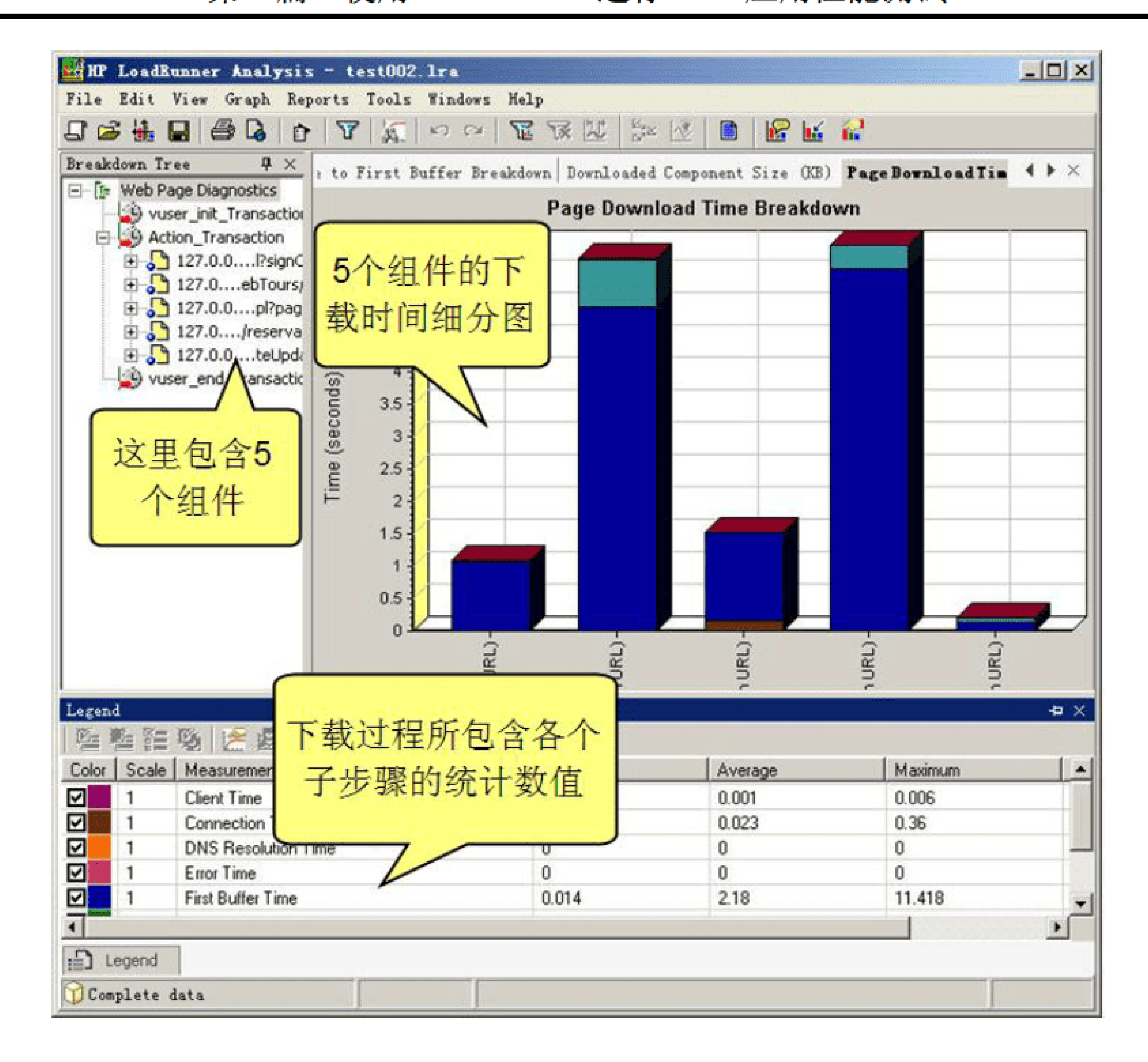


图 13-53　页面下载时间细分图

在图 13-53 左边的细分树视图中，Action_Transaction 事务包含了 5 个组件，因此右边的下载时间细分图则包含了 5 个柱体。每个柱体内部还包含下载过程中各个子步骤所花费的时间。这些子步骤按顺序如下。

- DNS 解析时间（DNS Resolution Time）：用户发送 HTTP 请求后，DNS 服务器将请求的 DNS 名称解析为 IP 地址并返回所需的时间。
- 连接时间（Connections Time）：与指定 URL 的 Web 服务器建立初始连接所花费的时间，如果该项数值较大，则需要考虑网络连接问题。
- 第一次缓冲时间（First Buffer Time）：与 Web 服务器连接后，用户发出请求到成功接收来自服务器的第一次缓冲所花费的时间。
- SSL 握手时间（SSL HandShaking Time）：建立 SSL 连接所花费的时间。它包括客户端呼叫、服务器呼叫、客户端与服务器端公钥证书传输等子步骤。当采用 HTTPS 协议进行通信时该项数值才不会为 0。
- 接收时间（Receive Time）：从服务器完成接收数据过程所花费的时间，字节数与时间比值可用于评估网络传输质量。
- FTP 验证时间（FTP Authentication Time）：该时间只有在采用 FTP 协议进行通信时数值才不为 0。显示了 FTP 服务器验证客户端所花费的时间。
- 客户端事件（Client Time）：客户端用户所花费的时间，包括在页面的思考时间等各种情况。
- 错误时间（Error Time）：从发出 HTTP 请求到返回错误信息所经过的平均时间。

在图 13-53 中，如果展开左边细分树视图中的某个组件，并双击其中的某个页面元素，还可以打开该页面元素的下载细分时间图，如图 13-54 所示。

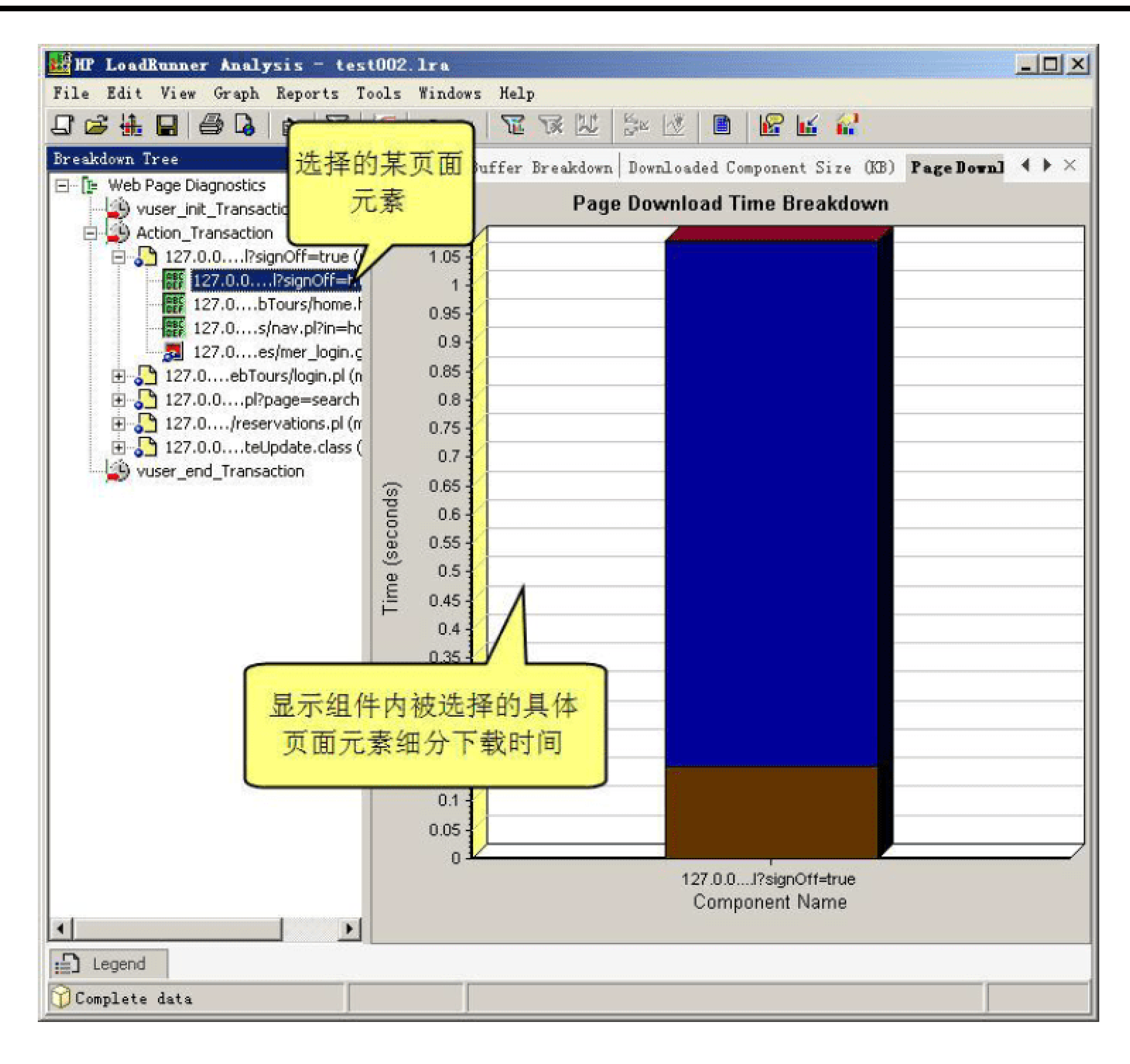


图 13-54　某具体页面元素的下载细分时间图

针对上述的各个时间，总体原则都是数值过大，则要考虑是否出现了异常。

3．页面调试图

页面调试图可以说是该类图表中最全面、综合的一个图表，它的界面也比较复杂，如图 13-55 所示。

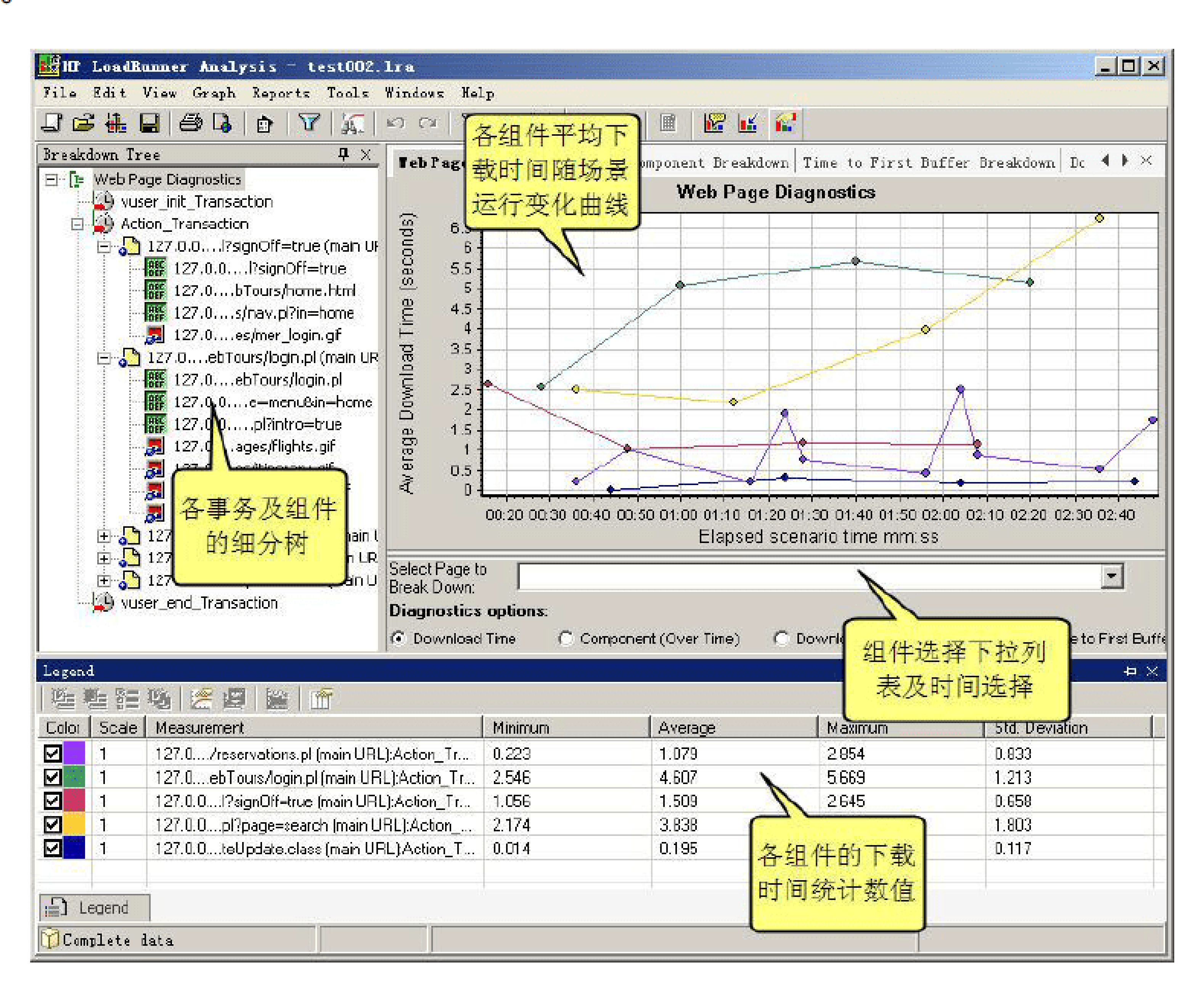


图 13-55　网页调试图

可以发现，图 13-55 左边的细分树视图中有 5 个组件，因此右边的图表视图中也有 5 条变化曲线。同时，在图表与图例的中间，还包含一个下拉列表框，其中列出了全部的 5 个组件 URL，性能测试工程师可以通过单击进行选择，从而改变图表中显示的内容。

在图 13-55 左边的细分树视图中，双击各组件名称，右边图表将显示该组件的下载时间随场景运行变化曲线图，同时下拉列表框将自动填充为所选择组件的 URL，如图 13-56 所示。更为方便的是，图 13-56 还显示了该组件所包含的各页面元素的下载细分时间表。

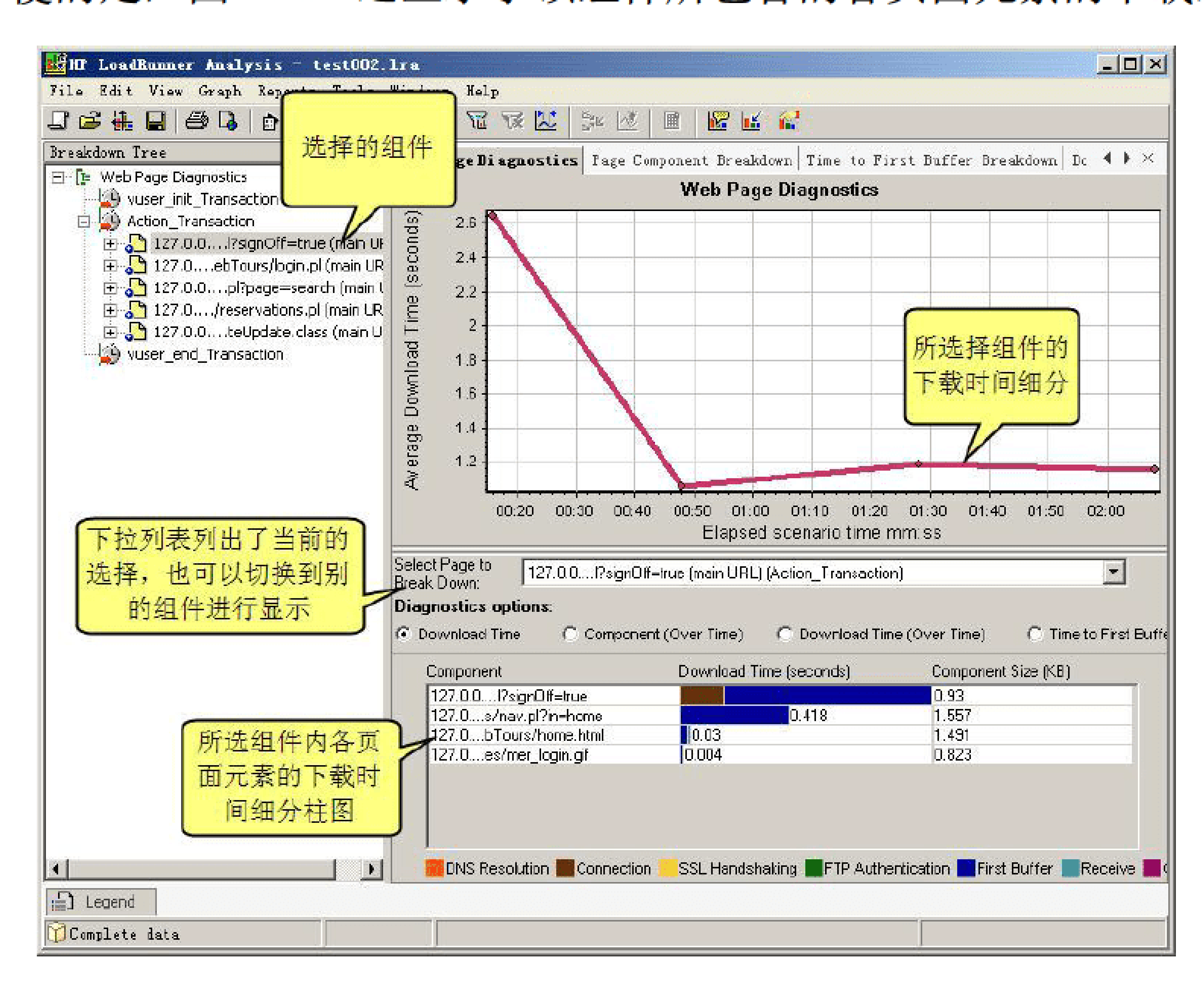


图 13-56　在网页调试图中显示某组件下载时间细分曲线

由于有了之前网页组件细分图的基础，这些较为复杂的图表理解起来也并不很困难。对于其他的网页调试图，感兴趣的读者可以在实际工作中使用并发现彼此之间的差别。

【网页调试图的作用】

使用网页调试图的主要目的就是提供了数据度量，用以发现性能问题是否由于网络问题、服务器问题还是 Web 应用的某一部分的问题所带来的。

13.3　本 章 小 结

本章重点介绍了 LoadRunner 分析器中有关 Web 应用的 4 类图表：

- ❑ 虚拟用户图；
- ❑ 事务图；
- ❑ 网络资源图；
- ❑ 网页调试图。

除此之外，还有一类系统资源图（System Resource Graphs）。它能为测试工程师提供 CPU 使用率、内存信息等多项基本性能指标数据，由于其添加方式与如上的几种图表基本一致，图表内容又为第 6 章所介绍各性能计数器的变化曲线，在本章就不详细介绍了。读

者务必记住该图对于分析性能问题是必备的。

对于这些图表本身，可以采取过滤、分组、下钻等技巧来获得图表中更为详细的信息。对于不同的图表之间，可以采取合并的方法来发现场景运行中性能的变化规律。对于不同场景的结果，可以采用交叉结果图的方法来发现多次测试之间性能的变化以及优化的效果。

常用于合并分析的图表组合有如下几对：

- 虚拟用户图与平均事务响应时间图，用于分析虚拟用户数量对于事务响应时间的影响。
- 每秒点击次数图与平均事务响应时间图，用于分析点击次数对于事务响应时间的影响。
- 吞吐量与平均事务响应时间图，用于分析吞吐量对于事务响应时间的影响。
- 网页调试图与平均事务响应时间图，用于分析性能问题是否与网络问题相关。

同时，还可以查看事务响应时间百分比图或者分布图来获得大部分事务响应时间数值的区间等。

通过本章讲解的各个图表，利用一些方法，不难得到当前本次性能测试的结论。截至目前，本书一直使用 LoadRunner 的分析器来进行性能测试结果分析，但在实际的工作中，性能测试工程师极有可能通过其他软件和手段来获得性能数据，因此学习一些通用的分析测试结果知识是很有必要的。另外，有了性能测试结论还不够，还需要将整个测试进行总结，形成性能测试报告发送给相关人员。这都将在第 14 章进行讲解。

第 4 篇　Web 性能测试提高篇

第 14 章　通用性能测试结果分析

截至第 13 章，小白已经完成了 LoadRunner 测试脚本的编写、场景的建立，并成功地执行测试及分析了整个测试结果、产生了测试报告。本书对于 LoadRunner 这一强大的工具也已经基本介绍完毕了。但是，在实际工作中，性能测试工程师可能并不一定采用 LoadRunner，而是使用其他的工具甚至自行编程来取得相关的性能测试数据。那么，有没有通用的一些分析性能测试数据的经验呢？

本章的内容就能够回答上述这个问题。如果说第 12 章中 LoadRunner 所提供的性能图表与测试报告是汽车中的自动挡，那么本章就是汽车中的手动挡。作为一名性能测试工程师，必须能够脱离工具软件的束缚，直接从原始的数据中得到正确的结论，才称得上合格。

初学者都知道，性能测试得到的数据，如果不进行分析，得到的结果是没有多大价值的。而在分析性能测试结果的过程中，往往需要用到简单的数学知识来进行评估、敏锐的观察能力来发现隐藏在数据中的性能问题；另外，出色的文档编写能力和图表制作能力则有利于将测试工程师了解到的事实有效地传递到相关人员的印象当中。其中，工作中最重要、也可能是最耗费时间与精力、同时又是收获最大的部分就是发现性能问题。

本章将依据以上这几种能力分为 3 节来介绍：第 1 节介绍判别测试数据是否可靠的要点，其中包含了一些简单的统计学知识。第 2 节通过几个范例来讲解发现性能的技巧与经验。第 3 节将列出编写性能测试报告的注意事项与要点，同时，对 Office Excel 软件中的画图功能也进行了较详细的说明。

14.1　性能测试结果的可靠性

性能测试的结果往往由大量数据组成，在我们拿到这些数据之后，首先要判断这些数据是否可靠。一些简单的统计学指标可以使我们很快地从数据中获得对于性能的基本印象。因此，本节先借助一些数学知识，让我们能够使用统计学上的简单指标对结果进行简单归纳；之后，再介绍判断性能测试结果可靠性的几条经验规则。

本节中将要介绍、同时也是测试结果分析中较为常用的数学指标有如下几种：

- 平均值（Mean Value）；
- 中值（Median Value）；
- 正常值（Normal Value）；
- 标准偏差（Standard Deviation）；
- 正态分布（Normal Distribution）；
- 一致分布（Uniform Distribution）；
- 置信区间（Confidence Intervals）。

为了理解方便，我们就使用小白获得的一手响应时间作为例子，学习如上数学指标的具体含义，从而获得总体性能的更直观印象。

14.1.1　原始数据

小白手中有关公司网站响应时间的数据一共分为 4 组，分别代表了客户端不同操作端系统、不同浏览器下访问首页的数据，分别如表 14-1、表 14-2、表 14-3、表 14-4 所示。

表 14-1　Windows XP下通过IE 6 访问公司网站的响应时间数据（秒）

5	6	4	8
13	10	4	4
5	7	8	10

表 14-2　Windows XP下通过Firefox 3 访问公司网站的响应时间数据（秒）

1	12	5	7
8	2	2	4
12	13	1	1

表 14-3　Windows Vista下通过IE 7 访问公司网站的响应时间数据（秒）

6	6	6	8
12	12	14	4
8	6	6	6

表 14-4　Windows Vista下通过Firefox 3 访问公司网站的响应时间数据（秒）

6	6	7	8
8	7	6	6
9	6	11	9

从以上数据来初步观察，响应时间长短参差不齐，表面上看分布也没有什么规律。

14.1.2　平均值

所谓平均值（Mean Value），就是把所有数值都相加，然后除以这些数值的个数。平均值也叫做算术平均数。平均值对于某类数据是非常有用的，比如考试成绩，年龄数据等。但是，在 14.1.1 所举出的数据来说，平均值并不能说明问题，甚至会隐藏可能的性能问题。

【实战演练：平均值的计算】

比如，在 14.1.1 的 4 个表格当中，就平均值来看，表现最好的应该是表 14-2。这是因为，表 14-1、表 14-2、表 14-3、表 14-4 的平均值依次为：

（5+6+4+8+13+10+4+4+5+7+8+10）/12 = 7；

（1+12+5+7+8+2+2+4+12+13+1+1）/12 = 5.67；

（6+6+6+8+12+12+14+4+8+6+6+6）/12 = 8.33；

（6+6+7+8+8+7+6+6+9+6+11+9）/12 = 7.41。

以上数值的单位均为秒。从结果中可以发现，表 14-2 响应时间的平均值确实是最短的。但是，需要注意的是，表 14-2 中的数据“两极分化”比较严重，既有很短的 1 秒，也有很长的 13 秒，而在平均值 5.67 秒附近的数值却很少。这样的情况就可能暴露出网站性能的某些问题，或者是数据采集中的不科学性。

读者在实际工作中得到测试结果的时候，不能单单记录数值，同时还要思考如下等问题：

- ❑ 响应时间很短是否由于缓存的缘故？
- ❑ 响应时间很长是否是 Web 应用代码的问题？
- ❑ 不同浏览器的响应时间有什么规律？

在测试过程中，对得到的数据多问几个为什么，有助于提高测试的准确性。

14.1.3　中值

中值的引入能够发现部分 14.1.2 节中表 14-2 中的数据问题。

【中值是什么】

所谓中值（Median Value），就是将数据从小到大排列起来，中间那个数的数值。

将中值的定义应用于实际，比如对于表 14-1 来说，其数据从小到大的排列依次为：

```
4 4 4 5 5 6 7 8 8 10 10 13
```

如果这一系列数据的个数为单数，中值就是中间那个数的值。如果类似表 14-1 的情况，数据个数是 12，为偶数，则中值一般认为是最中间两个数值（这里是第 6 个和第 7 个）的平均值，即（6+7）/2=6.5。

经过计算，表 14-1、表 14-2、表 14-3、表 14-4 的中值分别为 6.5、4.5、6 和 7。

【中值与平均值的关系】

中值虽然不等同于平均值，但若中值与平均值越接近，则说明数据分布的越均匀。

14.1.4　正常值

正常值（Normal Value）并不一定意味着它的值是正常的。所谓正常值，是指在数据结果中出现频率最多的那个值，通俗地说，就是在它们中间最容易碰到的数值。表 14-1、表 14-2、表 14-3 和表 14-4 中的正常值分别为 1、4、6、6 秒，可见，正常值与平均值、中值可能会有很大差别。

【实战演练：正常值的判断】

设想这样一个场景：小白所测试的 Web 应用有某部分代码出现了问题，导致数据库连接经常超时，测得的响应时间序列为 20、25、30、26、26、27、26 等，在这样的数值当中，26 是正常值，但它绝不是网站正常时应该具备的响应时间，因此，它又是“不正常”的。实际工作中这样的情况并不鲜见，读者在处理数据时一定要首先保证数据的有效性。

性能测试中正常值的意义在于发现当前配置下，多数情况采集到的数值是什么。

14.1.5　标准偏差

标准偏差提供了比中值更准确的方法，来确定数值是否“聚集”在平均值附近。标准偏差（Standard Deviation）是一种度量数据分布分散程度的标准，它可以用来衡量具体数据值偏离平均值（算术平均值）的程度。标准偏差越小，这些值偏离平均值就越少，数据就越可信。反之，数值偏离平均值越大，数据就越不可信。

【实战演练：Excel 计算标准偏差】

标准偏差的计算不是本书讲解的内容，我们只需理解它的含义会用工具计算就可以了。常见的计算标准偏差的工具软件就是微软的 Excel。如图 14-1 中新建了一个工作表，其中有 4 列，每一列分别包含表 14-1 到表 14-4 的测试数据。

下面举例计算前面表 14-1 中数据（已经录入在图 14-1 工作表的 A 列之中）的标准偏差。首先，单击工作表中的 A 列，然后选择“插入”|“函数”命令，如图 14-2 所示。之后 Excel 将弹出函数选择对话框，在其中找到计算标准偏差的 STDEV 函数即可，如图 14-3 所示。

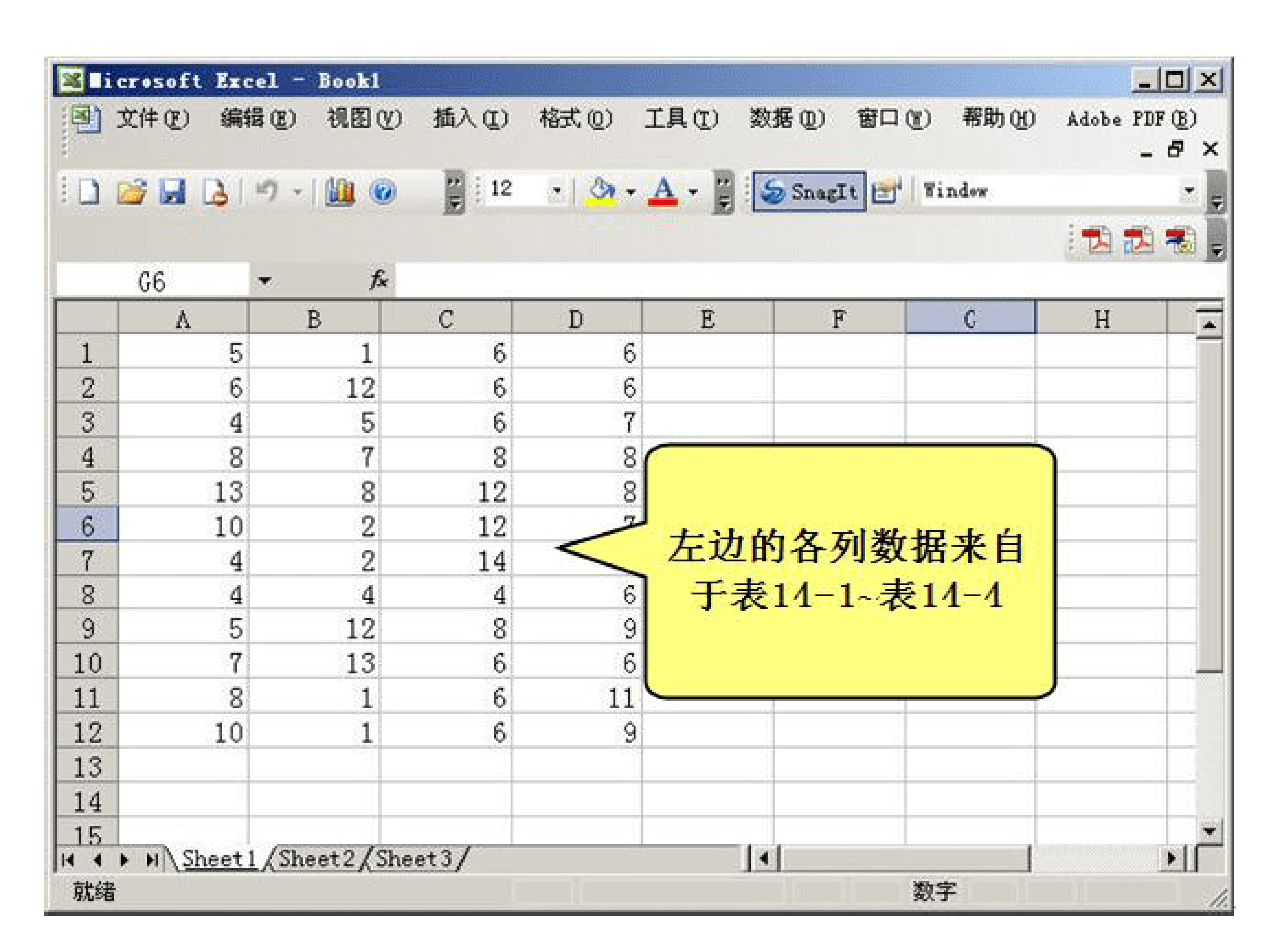


图 14-1　在 Excel 中新建工作表输入测试数据

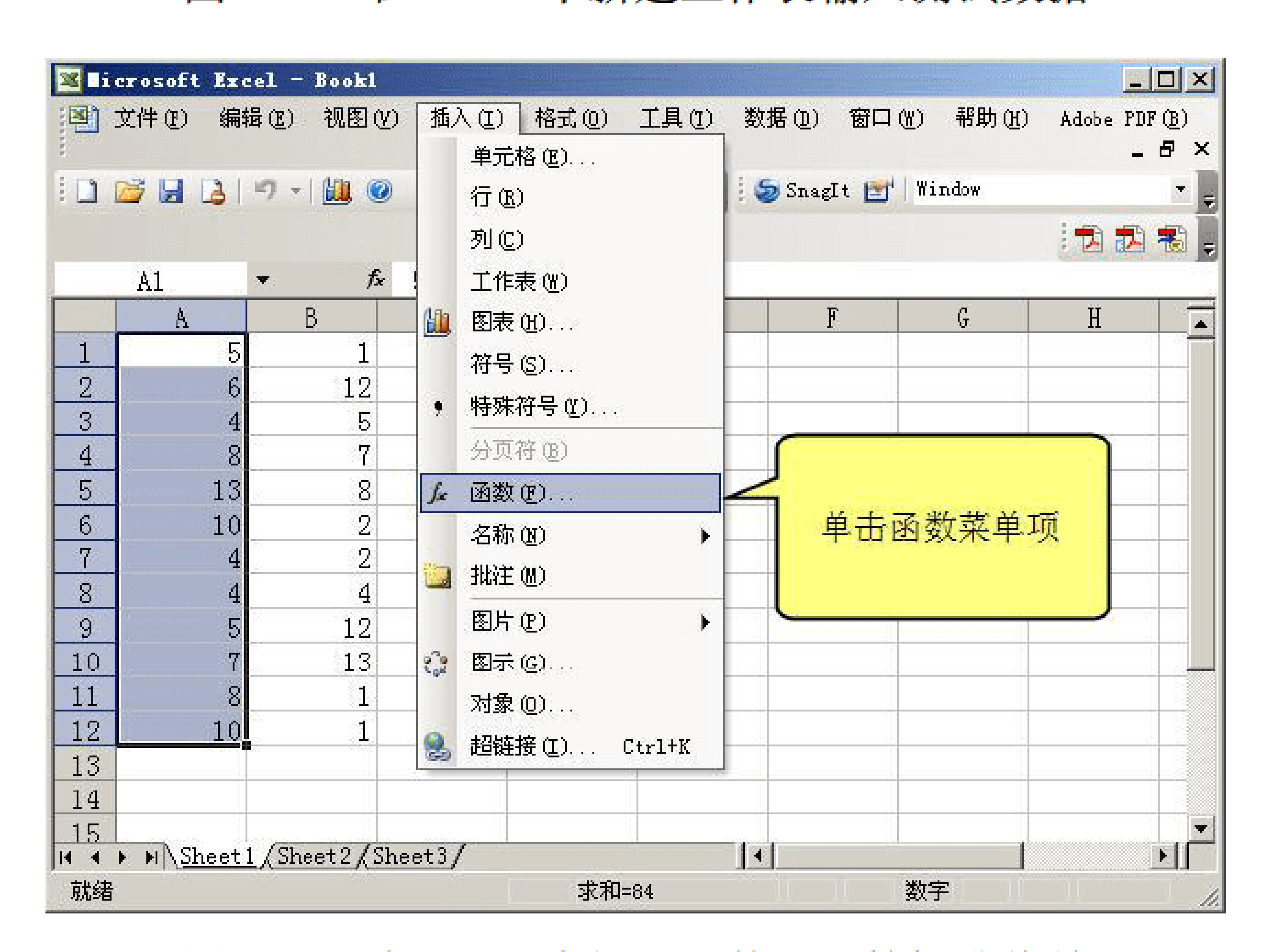


图 14-2　在 Excel 中插入函数以计算标准偏差

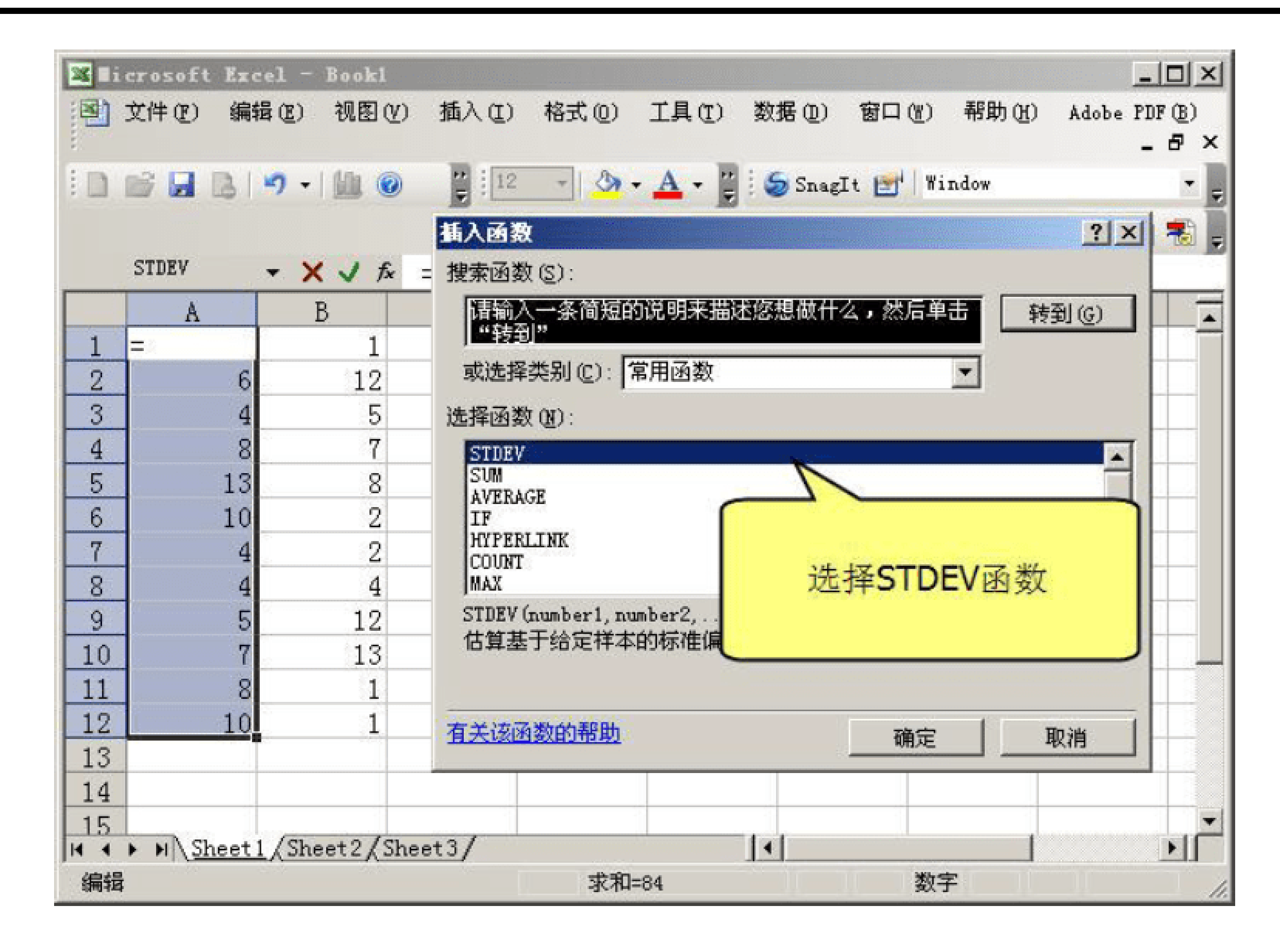


图 14-3　选择 STDEV 函数计算标准偏差

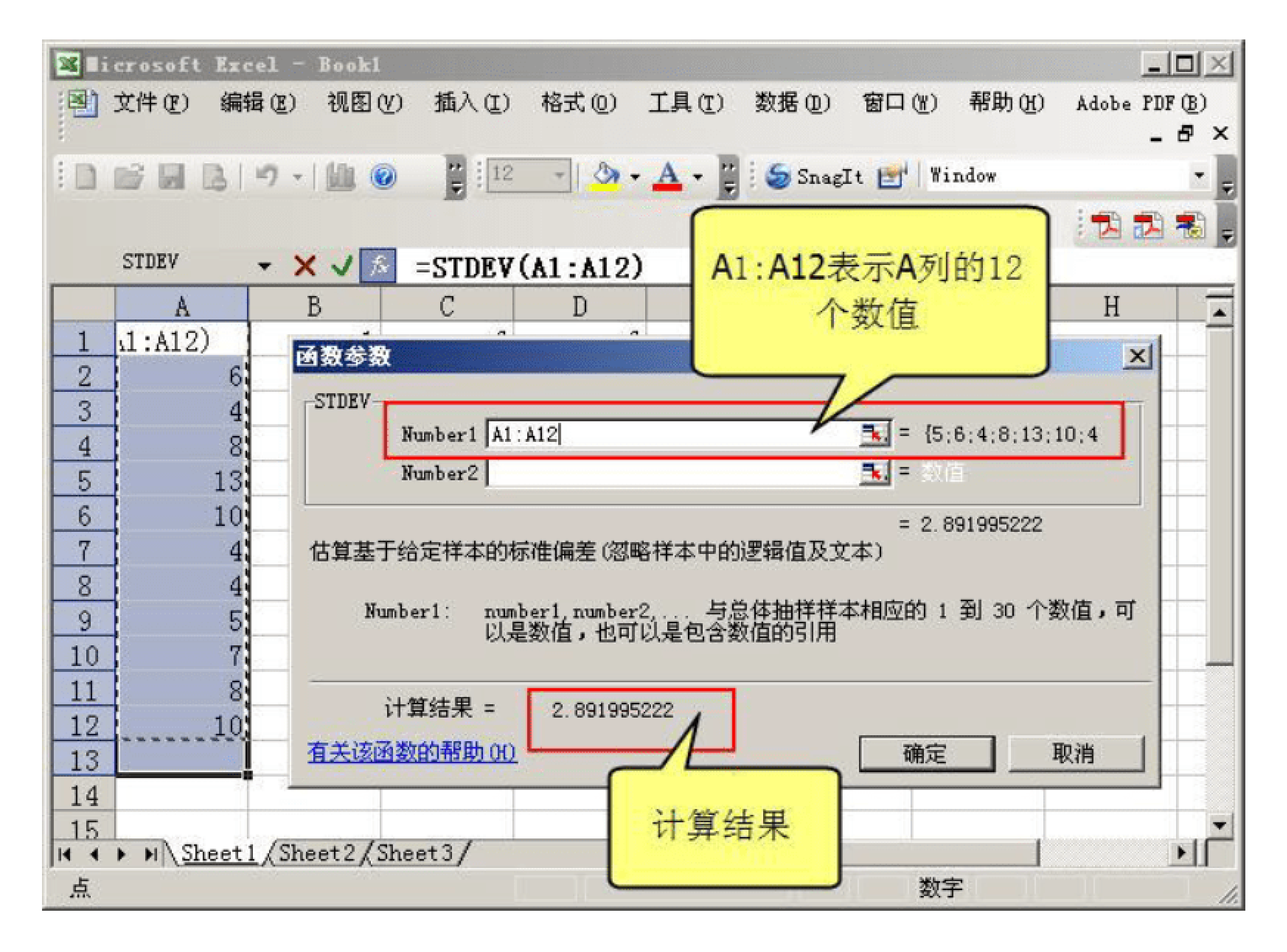


图 14-4　输入计算范围并获得标准偏差结果

计算标准偏差需要输入一系列的数值，这也是在图 14-2 中要首先选中 A 列或其他列的原因。在图 14-3 中单击“确定”按钮之后，Excel 将弹出函数参数对话框。如果其中的 Number1 文本框内没有数值，读者可以填入文字 A1:A12，以表示 A 列的所有 12 个数值作为计算的输入，依次类推。确认输入数据无误后，单击“确定”按钮，计算结果将立即显示在当前的对话框内，如图 14-4 所示。

通过前文这样的方法，依次可以得到从表 14-1 到表 14-4 所包含测试数据的标准偏差，分别为：2.8919、4.6384、3.1285 和 1.6213。根据前面所介绍的“标准偏差数值越小数据越值得信赖”原则，很显然，表 14-4 所包含的数据更具有可信度。

对测试结果进行标准偏差的计算，有利于从中发现更具可信度的度量数值。

14.1.6　正态分布

正态分布（Normal Distribution）也叫做钟形分布，这个名字是因为正态分布的数值在图形上类似一口钟而得来。它的含义就是一系列的数值当中，靠近中值附近的数值数量最多，而偏离中值的数值数量则不断减少。人类社会的很多行为都符合正态分布的特点，我们常说的“随大流”也可以说是一个体现吧：大多数人的行为都是非常类似的。

一个典型的正态分布图如图 14-5 所示。在性能测试产生的数据中，足够大量的响应时间具有正态分布的特点。

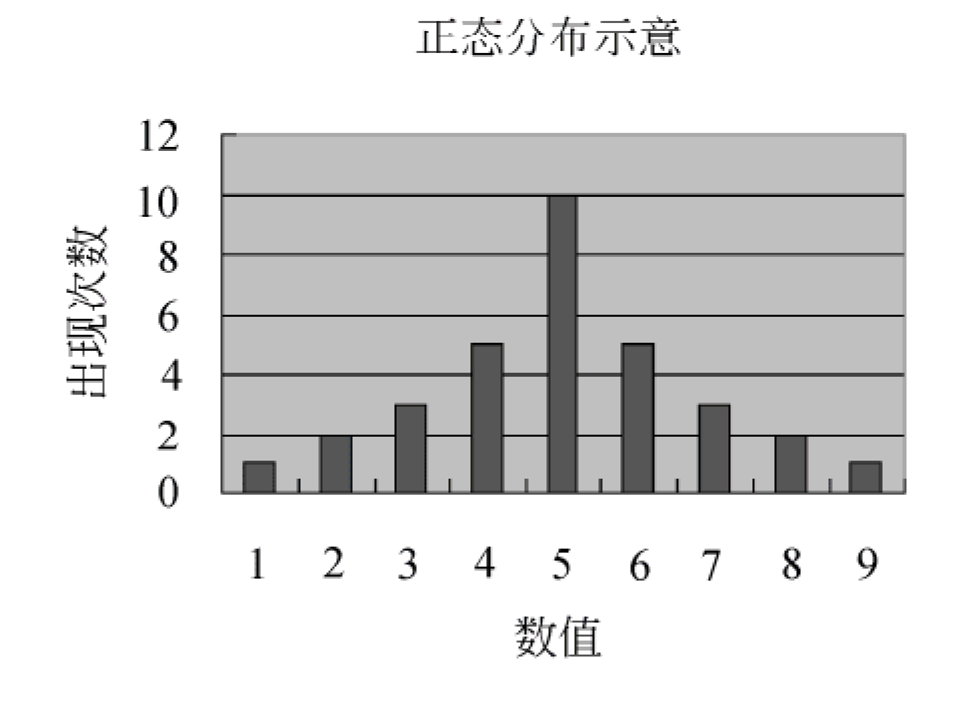


图 14-5　正态分布（钟形分布）示意图

【正态分布与标准偏差的关系】

正态分布与标准偏差有很大的关系，一般来说，标准偏差越小，数值越接近正态分布。因为正态分布存在非常普遍，所以才拥有了 Normal 这样的名字。

14.1.7　一致分布

一致分布（Uniform Distribution）顾名思义是指测试所取得的数据值相差很小，简单粗略地看，在图中会表现为波动很小的近似直线，如下面的情况。

【实战演练】

小白所在的公司每周要发送一个邮件列表给注册用户，该列表的内容实际上是一个由市场、销售部门 HTML 页面。由于发送程序运行在数据库服务器上（因为每周一次，也是周日晚间发送，所以暂时没有必要使用专门的服务器来完成），为了不显著影响整体性能，需要对 HTML 页面的大小进行限制。为此，小白记录了若干次的文件大小，如表 14-5 所示。

表 14-5　每周邮件列表文件大小

日　期	文 件 大 小
2008 年 9 月 6 日	47KB
2008 年 9 月 13 日	48KB
2008 年 9 月 20 日	47KB
2008 年 9 月 27 日	48KB

如果在 Excel 中对表 14-5 所列出的数据画成图，就可以看成是一致分布，如图 14-6 所示。

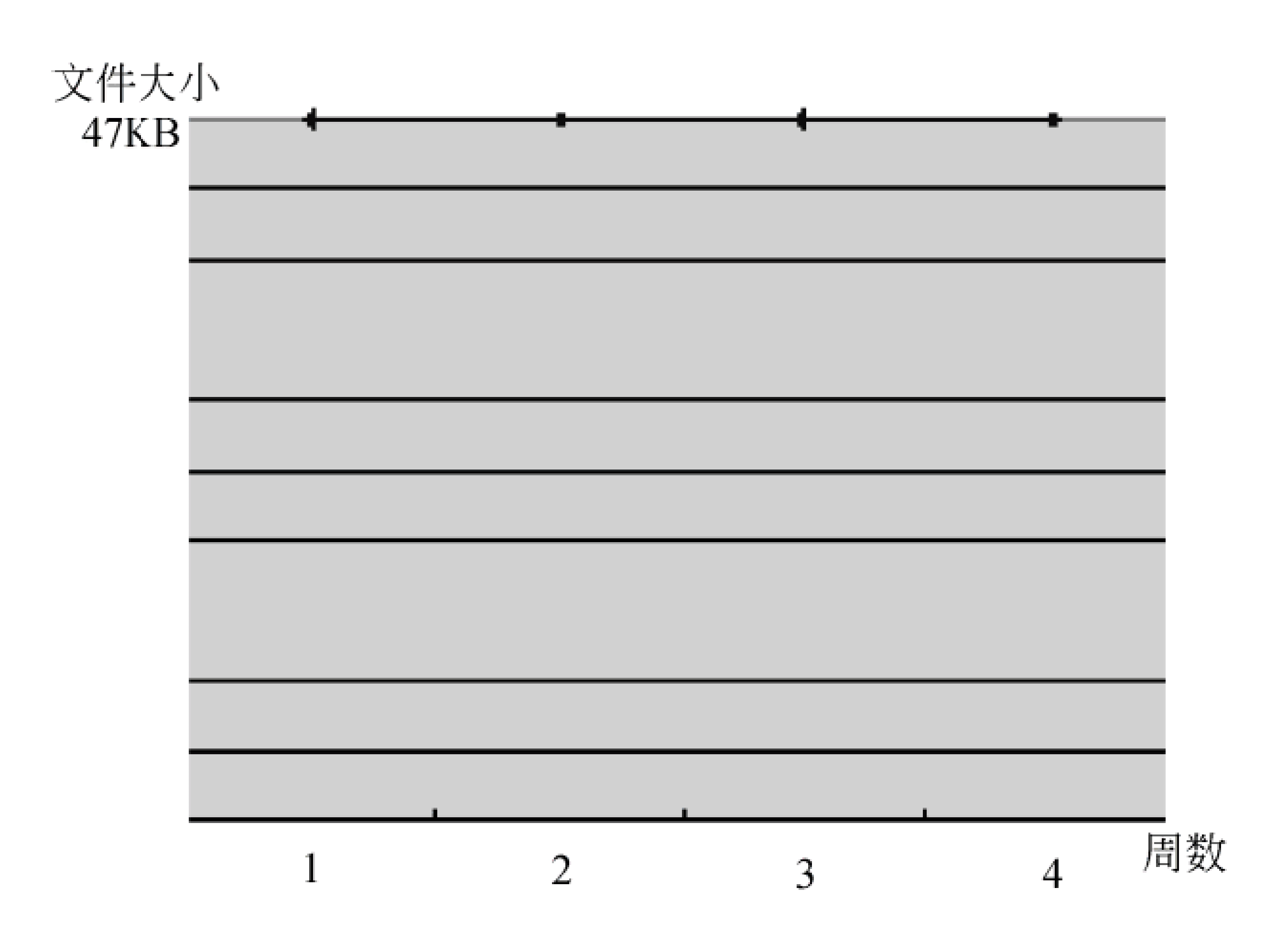


图 14-6　邮件列表内容文件大小呈一致分布

每次邮件列表大小基本一致，是因为市场、销售部充分利用了文件大小的上限，尽量争取在有限的大小之内，放入更多的宣传内容。当然，在实际工作情形中，不一定每次都会出现这样的情况。

如果在性能测试中出现了一致分布的数据，测试工程师需要找出原因，一般来说，这样的数据反而是值得怀疑的。比如响应时间，如果用户的响应时间惊人的一致，则要考虑是否有部分用户因为某些原因根本无法访问网站等原因。

14.1.8　置信度与置信区间

在性能测试领域，置信度指的是测试结果与真实结果之间的差别。由于具体的测试结果是由用户使用 Web 应用方式的估计模型和性能测试方法决定的，因此也可以认为置信度反映了网站人员与最终用户在使用该 Web 应用上的相似度。

【置信度举例】

举例来说，小白所在公司的人都认为用户将更喜欢网站的 A 栏目，因此在资源有限的情况下，测试部对该栏目的功能进行了重点性能测试，并进行了优化，获得了不错的结果；但实际网站上线后，用户却更喜欢另外某个栏目，经常使用的功能也与事先预想的不同。这就会导致性能测试结果与实际性能测量值有所误差。这种误差大小的程度就是置信度。当然，这样的理解在数理统计方面并不严格，但对于性能测试工程师在工作中的使用已经可以了。

置信度越高，置信区间（Confidence Interval）也就越接近真实值的范围。置信区间是指在某一置信度水平下，性能测试结果与 Web 应用上线后实际运行结果间的误差范围。要知道在 Web 应用上线前，没有谁能准确地预计用户行为，因此有必要在进行性能测试时预估一个置信度，再根据结果得到置信区间。

【置信区间的实际使用】

假设公司对网站响应时间设置的合理值为 10 秒以下，置信度估计为 80%。小白在对网站使用 LoadRunner 进行并发测试后，发现：由于使用页面功能不同，最差的情况一次并发 50 个用户就可能令第 51 个用户响应时间超标；而最好的情况则是一次并发 300 个用户才能令响应时间超标。那么，在测试结果报告中，小白应当这样进行陈述：根据 80%置信度，在一般工作负荷下，并发数为 50×80%到 300×80%，即 40～240 个用户，都不会引起响应时间超标。这里的 40～240 就是置信区间。

综上所述，置信度也可以理解为一种形式的安全系数。

14.1.9　数据可靠性判断的规则

前面讲述了一些统计学的知识，实际目的都是为了使得我们的性能测试报告能够更接近于真实，这样才能发挥最大的作用。因此，在测试结果出来之后，并不要立刻发送测试报告，而是要先判断取得的测试数据是否可靠，这样的能力对于性能测试工程师来说是非常必要的。

数据可靠性有如下几条经验规则：

（1）如果有超过 20%的测试数据明显与其他数据有很大差别，则应该先检查测试过程中是否出现问题。这样的情况是经常发生的：小白使用很多台测试机器在下班后运行自动访问公司网站的脚本程序，从而记录响应时间等数据。但由于其中某些机器设置了 Windows 系统默认凌晨 3 点发生的自动更新，可能会强迫重启电脑，从而导致测试中断，在重启过程中取得的响应时间数据当然是不可靠的。

（2）如果进行了多次相同目的的性能测试，如果某一次测试绝大多数结果比其他几次测试中最大的结果都要大，或者比它们当中最小的结果都要小，那么应该考虑这次测试结果的有效性问题。这一点是很好理解的。至于绝大多数的比例设定为多少，可以根据实际情况来定，一般至少在 75%以上。

根据以上这两条基本原则，再结合具体被测试软件的实际情况，就可以判断出哪些数据是可靠的，哪些数据是不可靠的。有了可靠的数据，才能编写出可靠的测试报告，这是最重要的一点。

在 14.2 节，本书将简要介绍性能测试结果的分析方法，这是性能测试报告的关键。

14.2　性能测试结果分析方法

在 14.1 节中我们学习了对于性能测试数据有效性的判断，并且利用统计学知识与以往经验从中可以得到性能概况。另外，还可以根据几条经验规则对怀疑数据进行剔除。在本节中将讲解如何对现有有效的数据进行分析，并据此得到性能好与坏的结论，为编写性能测试报告做准备。

对性能测试结果进行分析需要依次进行以下几个步骤：

（1）判断影响性能的因素。

（2）运用隔离、对比等方法进行趋势判断。

（3）记录各个结果，发现规律。

14.2.1　判断影响性能的因素

判断影响 Web 应用性能的因素其实一般发生在性能测试的设计阶段，但列在这里也是适当的，因为：

- ❑ 在实际工作中，设计阶段可能对于影响因素考虑不足，导致有些因素没有考虑到。这种情况会令测试结果不完全，要增加测试的运行。
- ❑ 设计阶段也可能出现考虑影响因素过多的情况，这需要在测试数据中进行判断，从而将无关的因素去除。该情况下那些多余的测试结果并非没有益处，因为正是它们，使得性能测试工程师得以发现了无关因素。

在 Web 应用领域，影响性能的因素有用户数量、服务器性能（请见前面章节有关性能计数器的部分）、网络带宽、客户端软件配置等多种。由于这些因素是综合起作用的，它们对 Web 应用的影响可以组合出很多种情况，其中部分如表 14-6 所示。

表 14-6　影响性能各因素的部分组合

情况编号	用 户 数 量	服务器性能	网 络 带 宽	客户端软件配置
1	变化	不变	不变	不变
2	变化	不变	变化	变化
3	变化	不变	不变	变化

可以知道，这样的组合还有很多种，如果都针对它们进行测试，在大多数情况下（时间、人力、软件发布要求等要求比较严格紧迫）是不可能完成的任务。因此，有必要使用隔离的方法进行精简。

不过，值得一提的是，对性能做出正确判断需要足够多的测试数据。

14.2.2　隔离与对比

隔离、对比是常见的生成、分析数据的方法。通俗地讲，隔离就是指固定其他的影响因素，只变化剩余那个影响因素的方法。对于表 14-6 的情况来说，我们只需要测试 4 种情况就可以了，即：

- ❑ 用户数量变化，其他因素不变的情况。
- ❑ 服务器性能变化，其他因素不变的情况。
- ❑ 网络带宽变化，其他因素不变的情况。
- ❑ 客户端软件配置变化，其他因素不变的情况。

这样一来，问题得到了简化，同时，各因素的影响规律也能够被发现。

并列是分析数据的方法，特别适用于将数据生成为图表的情况下。它可以分为纵向比较与横向比较。

【纵向比较】

所谓纵向比较就是在同一个影响因素数值变化、其他因素固定的情况下，将多次测试

结果并列在一张图表当中进行分析，从而发现该因素对性能的影响规律。比如针对表 14-6 中的数据，可以将用户数量分别是 100、1000、10000 的时候制表进行比较，发现用户数量对于性能的影响规律。

【横向比较】

而横向比较则是将多次纵向比较的结果，并列在一张图表中，从中发现各因素之间的关系或者性能的变化趋势。这多用于判断性能的优化成果。比如针对表 14-6 中的数据，可以在优化前与优化后进行不同用户数量的测试，如果测试结果曲线变化明显不同，则可说明优化的效果好坏。

14.2.3　详实记录中间结论

详实记录中间结论对于分析性能测试数据是非常重要的。实际工作中，经常发生初始的结论与最终结论不一致的情况。在分析每张数据表格或者图之后，如果可能，我们都要记录下该图或者表格说明了什么问题，有什么疑问。通过这样的方式，测试工程师对于整个 Web 应用的性能图景会逐渐明晰，也有利于做出错误结论后的回溯，发现分析思路上的错误。

总之，对于性能测试结果的分析，要有认真负责的态度和细致科学的方法。有了它们，不难得出正确的结论。

14.3　性能测试报告编写技巧

在对结果进行分析并得出结论之后，性能测试工程师要把它们以文字报告的形式发送给相关人员。这就是性能测试报告。除了书面文字之外，可能的话，公司还会召集人员开专门的会议进行报告讲解和结果分析。所以，性能测试报告是性能测试工程师的工作成果，也是公司其他部门考察性能测试工程师能力的重要窗口，编写出一份优秀的报告对公司的决策以及个人的职业生涯都非常有益处。

14.3.1　什么是好的性能测试报告

实际工作中的性能测试报告，一般是以 Word/PDF 格式文档或者电子邮件形式存在。而测试报告的读者，一般是整个项目组的管理者甚至更高层面、相关同事比如开发人员等，他们并不一定具备多少测试背景知识，因此，测试报告要尽量避免测试术语，要用容易理解的话语进行叙述。另外，它不应该是性能测试结果的简单罗列：因为读者是上级或者其他同事，他们没有多少时间来关心测试的具体细节，而只关心报告中测试结论是否合理以及结论的内容。这是需要性能测试工程师注意的原则问题，即不能从自己出发来写报告，而应该为报告的读者考虑。

根据这样的原则，要完成一份好的性能测试报告，最好做到如下几点：

- ❑ 提交报告的时机。
- ❑ 可以与测试主管就报告进行讨论。

- ❑ 有效地总结概括测试数据。
- ❑ 报告应该清楚易读，结合图表，但不能滥用图表。
- ❑ 报告要具备较强的逻辑性。
- ❑ 报告要具有层次感，几个部分区分明显、清楚。

测试报告一般分为测试目的、测试方法、测试数据概括总结、测试结果分析、结论这几大部分。在实际工作中的要求不尽相同，有的公司会有自己的模板，因此在文档结构上并无一定之规。但内容方面，如果能做到如上几点，编写出一份很好的性能测试报告就不是困难的。

在下面几节中，本书将一一介绍如上几个规则。

14.3.2　提交报告时机

与功能测试等不同，性能测试在整个 Web 应用的开发过程中并不是连续进行的，因此性能测试报告一般只会在几个时间点附近（比如某阶段结束前）才能让有限的读者看到。这容易给人以一种印象，性能测试并不如功能测试那样重要，如果时间紧迫，甚至不用很系统地进行。这就会影响到性能测试工程师的成就感和积极性。因此，性能测试工程师有必要创建一种性能测试持续存在的氛围。这样做有如下几个益处：

（1）培养同事对于性能测试的关注，普及性能测试的一些知识。这有助于测试报告的读者更好地理解性能测试的过程与测试报告的内容。由于日常工作中主动介绍性能测试知识显得比较突兀，根据实际情况，可以选择在测试部门会议之中选取短暂的时间介绍一些性能测试的理念。

（2）利于项目组内团结协作精神的培养。分享自己可以使人获得更多。比如，在开发人员刚刚修改完一些代码的时候，性能测试工程师不妨做一次小小的测试，如果比之前性能有所改进，就可以将结果用电子邮件的方式，不那么正式地发出来，同时还可以将性能测试数据放置于项目组内的服务器之中共享给所有成员。通过这样的方式，开发人员可以得到性能改善的好消息，互相鼓励，性能测试工程师的工作也让整个项目组看到，喜欢钻研的同事还可以到服务器共享中查看数据，性能测试工程师或许就能获得更好的反馈。

总之，提交报告的时机需要掌握，有如下技巧：

- ❑ 正式的性能测试结束后，要尽快发送整理好的测试报告，供决策、优化之用，以体现效率。
- ❑ 在两次正式性能测试之间，可以执行若干轻量级的性能测试，将改善的地方非正式地通知全组，以普及性能测试常识、激励同事与自己，提高团队精神。

14.3.3　与测试主管的讨论

前文提到，性能测试报告的读者是其他同事、部门甚至更高级别的领导，因此在发送报告之前，有必要与测试主管就报告内容进行讨论。通过讨论，至少可以获得如下的信息：

（1）发现问题。测试主管一般来说测试经验更为丰富，遇到和解决过的问题较多，因此他/她可能会发现现有报告的问题，进而提出改进的意见。

（2）使得报告表达更清晰易懂。总体说来，测试主管与其他同事、部门乃至更高级别

领导沟通机会较多，对于报告潜在的读者了解更深入，熟悉他们的阅读习惯与表达方式。如果有了更详实的读者信息，那么报告做有针对性的修改，会更清晰易懂。

（3）增加工作交流的机会。在一个团队当中，信息共享是很重要的，与测试主管的讨论有助于主管了解当前的工作，可以为性能测试工程师解决一些困难。

总之，多与测试主管进行工作上的讨论，对于一名初级性能测试工程师的成长是很有裨益的。

14.3.4　有效总结测试数据

有效地总结测试数据包含如下几个要点：

（1）在测试报告的内容中，测试数据不能分散在各个部分当中，而应该单独列为文档的某一部分。这样的安排可以使得文档结构更加清晰，读者在阅读测试数据的时候更加专注数据本身。

（2）对于测试数据，不可能将所有的数据都列于测试报告之中，可以将最能支持结论的数据列出一行并说明各数值的代表含义。同时，必须列出获取测试数据的方法，用尽可能简单的语言陈述清楚。

对于报告中的测试数据，我们需要掌握的原则就是它必须真实，并且能够有力地支持结论。测试数据与测试方法部分一般放置于报告的前半部分。

14.3.5　测试报告与图表的结合

前文多次提到测试报告要清晰易读，而图表就是增强可读性的一种有效方式。对于枯燥的数据来说，人们很难从数字中快速发现规律和趋势，而一旦将数据转换成图表，情况则会明显不同，趋势往往很直白。另外，颜色搭配合适的图表相对更容易吸引读者的关注。

【图表的副作用】

但是，需要注意的是，图表不是越多越好，因为那样的话单个图表所能吸引的关注就会被平均分配，最后降低到文字的水平甚至更低，这里边或许也有所谓“审美疲劳”的因素。因此，图表不在于多，在于精，在于支持结论，说明问题。

在 Excel 中将数据转换为图是非常容易的，14.3.6 节中我们将介绍这样的方法。

14.3.6　在 Excel 中为数据生成图

在 Excel 中为数据生成图表的方法实际上利用了 Office Web Component 组件的功能，它默认是与 Office 一起安装在硬盘中的。如果在控制面板中没有发现该组件，可以在微软官方网站免费下载。

下面笔者通过简单的步骤，为表 14-7 中某 Web 应用各栏目首页的 6 次响应时间数据生成图。

表 14-7　某次性能测试的响应时间数据

请求频道首页	响应时间（秒）					
网站首页	7.3	6.8	7.1	6.9	7.0	8.5
新闻栏目首页	8.5	8.4	8.0	7.8	7.9	8.3

续表

请求频道首页	响应时间（秒）					
论坛栏目首页	9.8	9.5	9.4	9.0	9.4	9.6
联系我们栏目首页	3.2	3.2	3.3	3.2	3.2	3.3
产品栏目首页	4.5	4.6	4.3	4.5	5.0	5.0

将上述的数据输入 Excel 的一个工作表，并全部选择数据与说明的单元格，如图 14-7 所示。

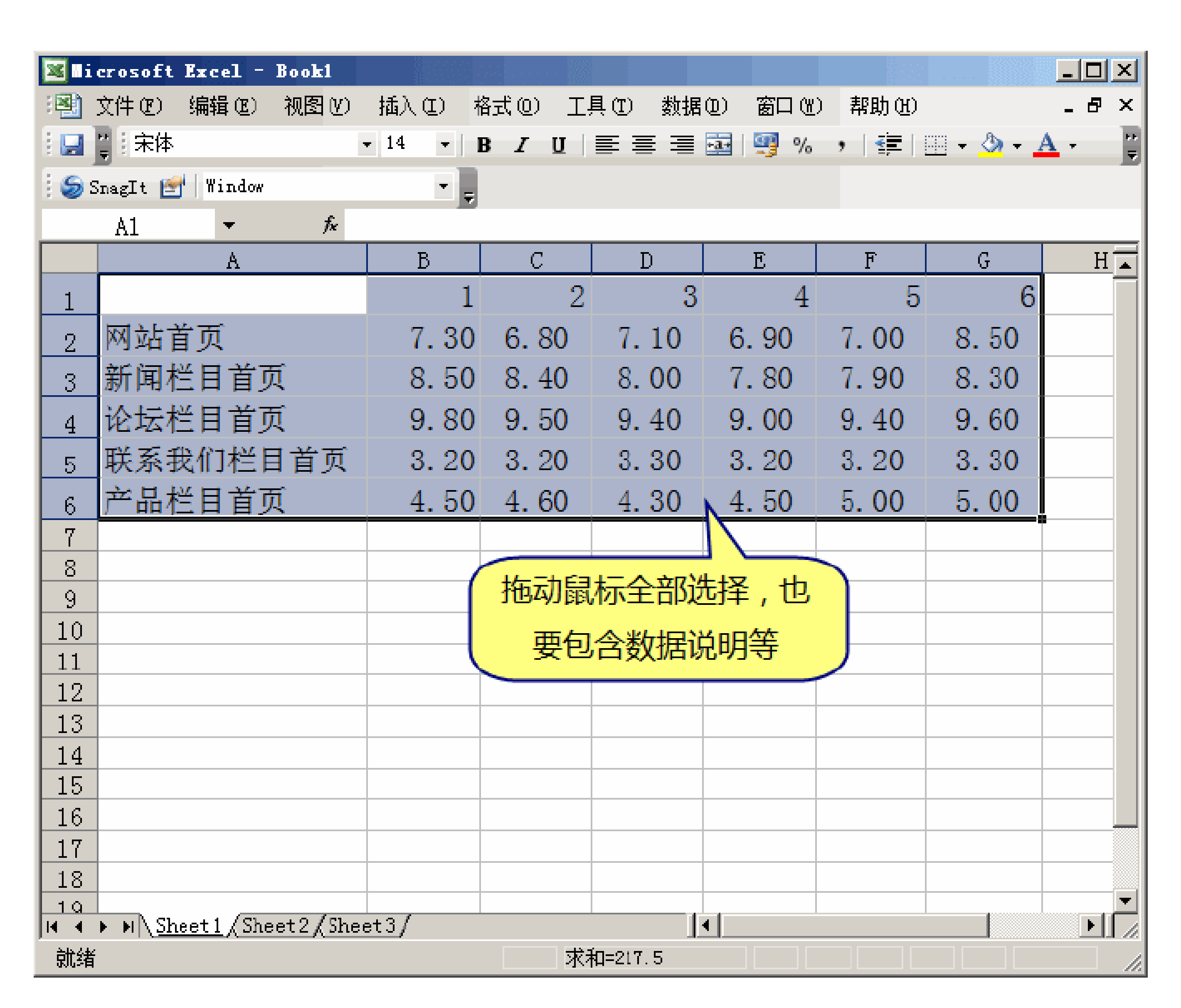


图 14-7　在 Excel 中选择待生成图的数据表

【其他数据文件格式与 Excel】

在实际工作中，测试结果绝大多数情况并不是 Excel 默认的 xls 文件，而以 txt、log、csv 为后缀名的文本文件居多。这是因为，这些文件没有附加的格式信息，生成容易并且相对快速。那么，这些文件中的数据是否也可以在 Excel 中生成图表呢？答案是肯定的。利用 Excel 的“文件”|“打开”菜单，选择打开文件类型为“文本文件”，就可以将很多使用 Tab 键、空格或者逗号对测试数据进行分隔的文本文件导入到 Excel 的工作表中，非常方便。

选择完毕后选择“插入”|“图表”菜单，即可以打开生成图表的向导，如图 14-8 所示。另外，在导航菜单中也可以单击图 14-9 中说明文字指向的“图表向导”（Graph）图标按钮达到相同的目的。

图表向导由 4 个步骤的设置对话框所组成，第 1 步需要确定的是图表的类型，界面如图 14-10 所示。

这里列出了日常工作中应用的绝大多数类型，如果有需要，也可以自定义新的图表类型。

【图表类型的选择】

不同类型的图表，适合于表现不同的数据关系。因此，在报告中应用正确的图表有利于表达与支持报告的结论。在性能测试领域，一般来说折线图适合表达响应时间等数值越

小越好的数据；柱形图、条形图适合表达并发用户等数值越大越好的数据；饼图适合表达 Web 应用终端用户信息，比如浏览器版本、IP 分布等；XY 散点图适合表达整个领域各组成元素的同一类数据等。

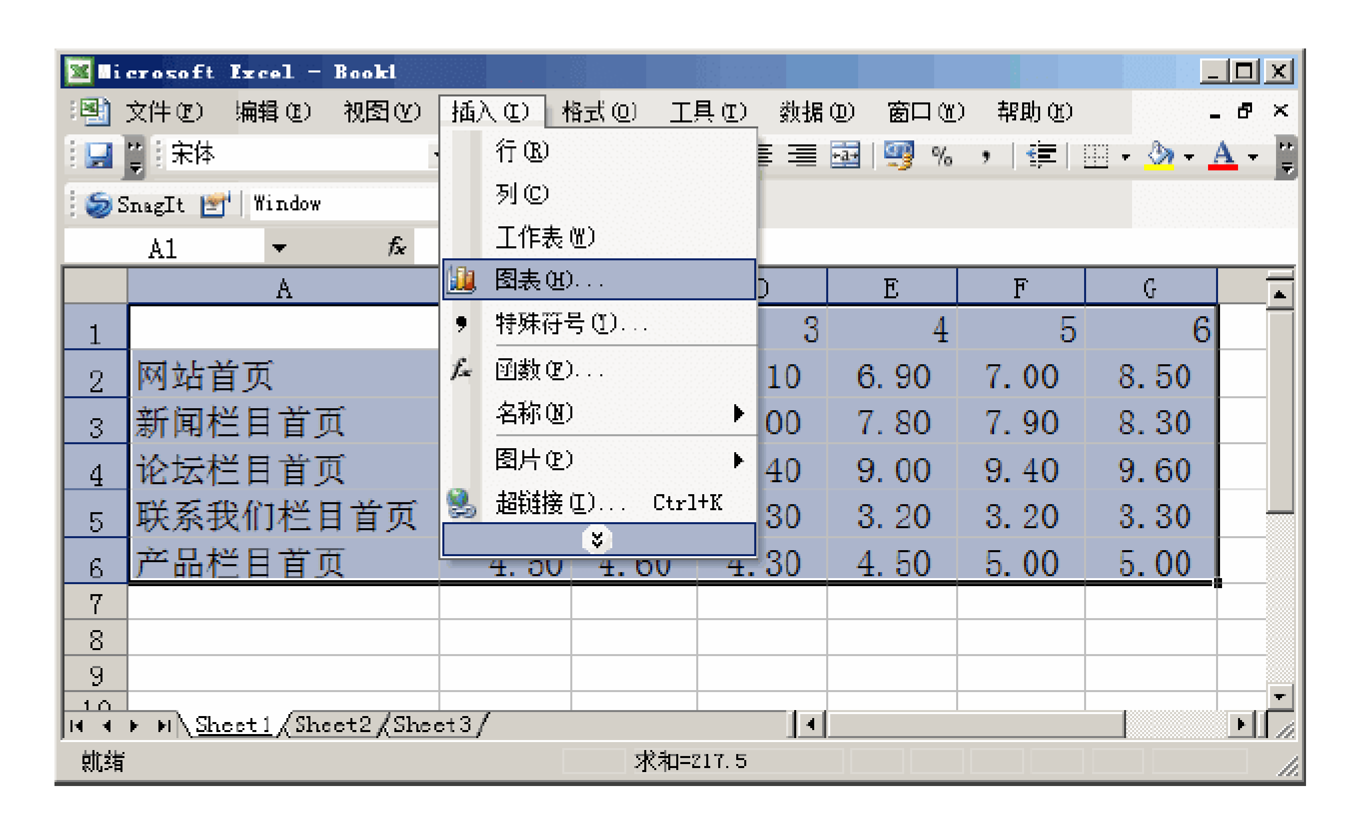


图 14-8　通过菜单打开图表向导

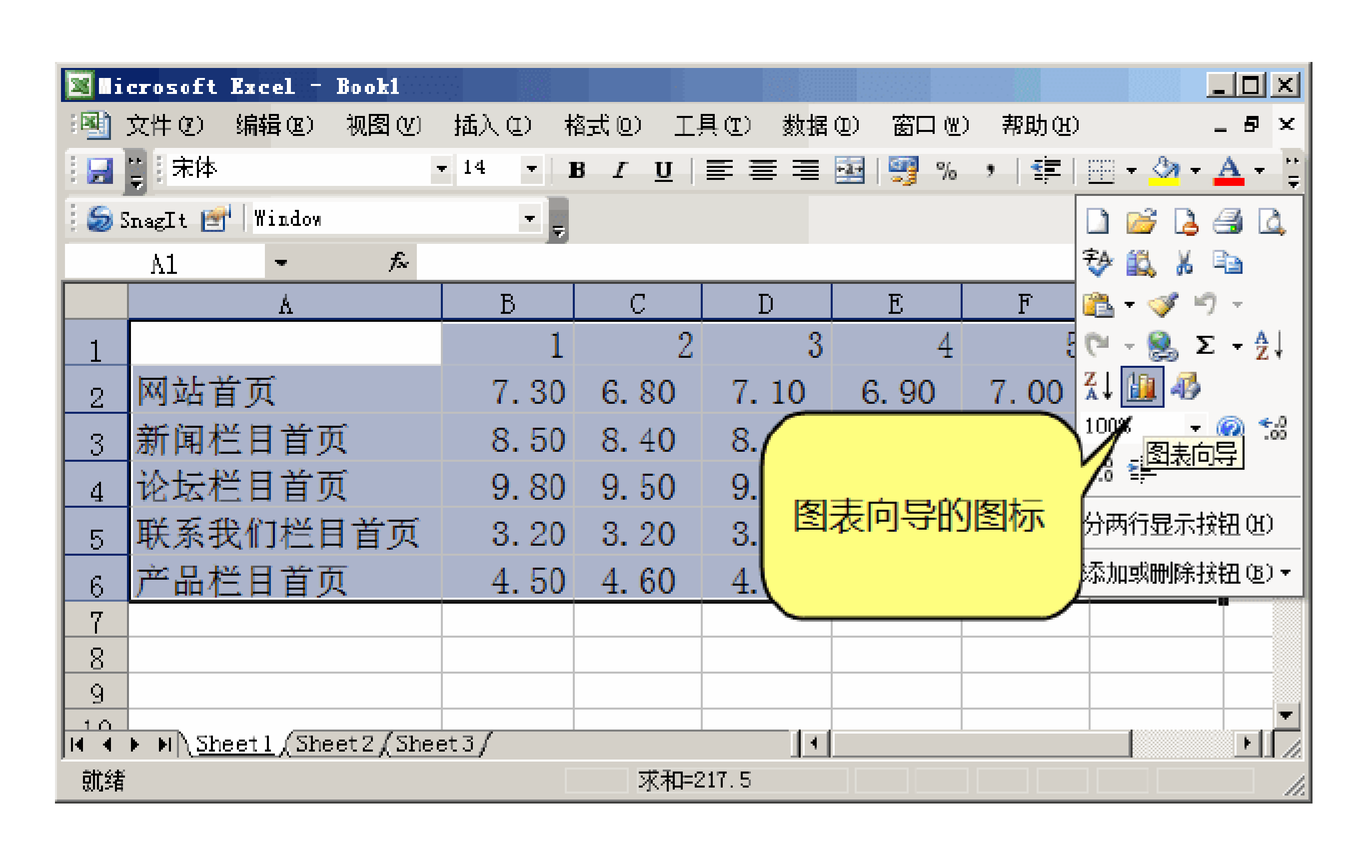


图 14-9　通过导航条图标打开图表向导



图 14-10　图表向导第 1 步：选择类型

在本节，笔者选择折线图来表现表 14-6 中的数据，因此，在图 14-10 中“图表类型”列表框中选择“折线图”，并在右边的“子图表类型”选项区域中选择默认的数据点折线图，单击“下一步”按钮，对话框显示界面如图 14-11 所示。

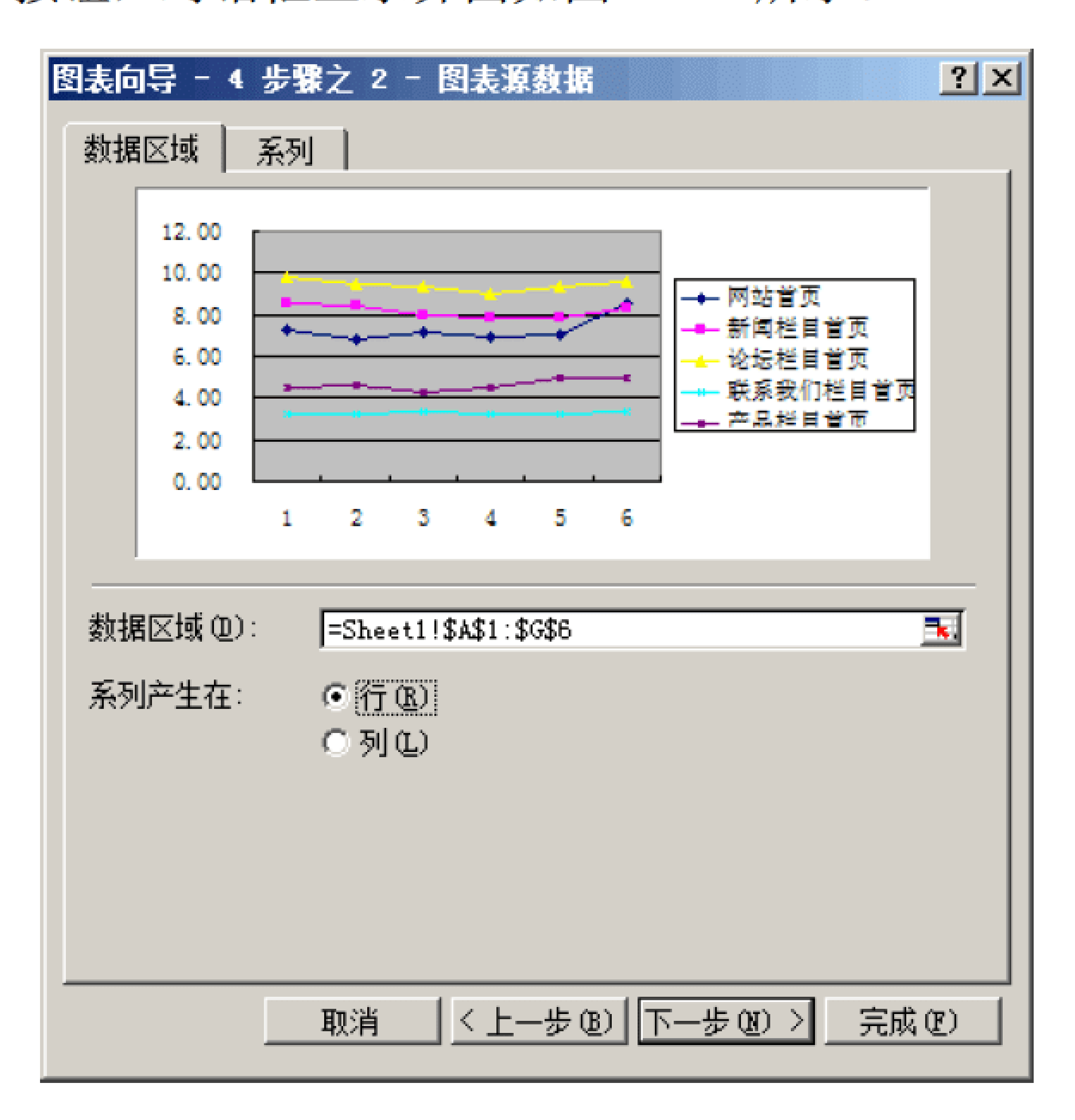


图 14-11　图表源数据的选择

图 14-11 中的对话框需要使用者选择哪些数据用于生成图表，由于在前面的步骤中，我们已经将数据和说明选择好，因此这里一般不需要做改变。另外，通过单击“系列”标签，可以进一步增添或者删除数据系列(在此处的例子当中，就是图 14-7 中的任 1 行数据)。

单击“下一步”按钮，进入图表向导的第 3 个步骤：设置图表选项。在图 14-12 所示的对话框中，可以对图表上额外的说明信息进行自定义，其中用处较大的有如下几个设置。

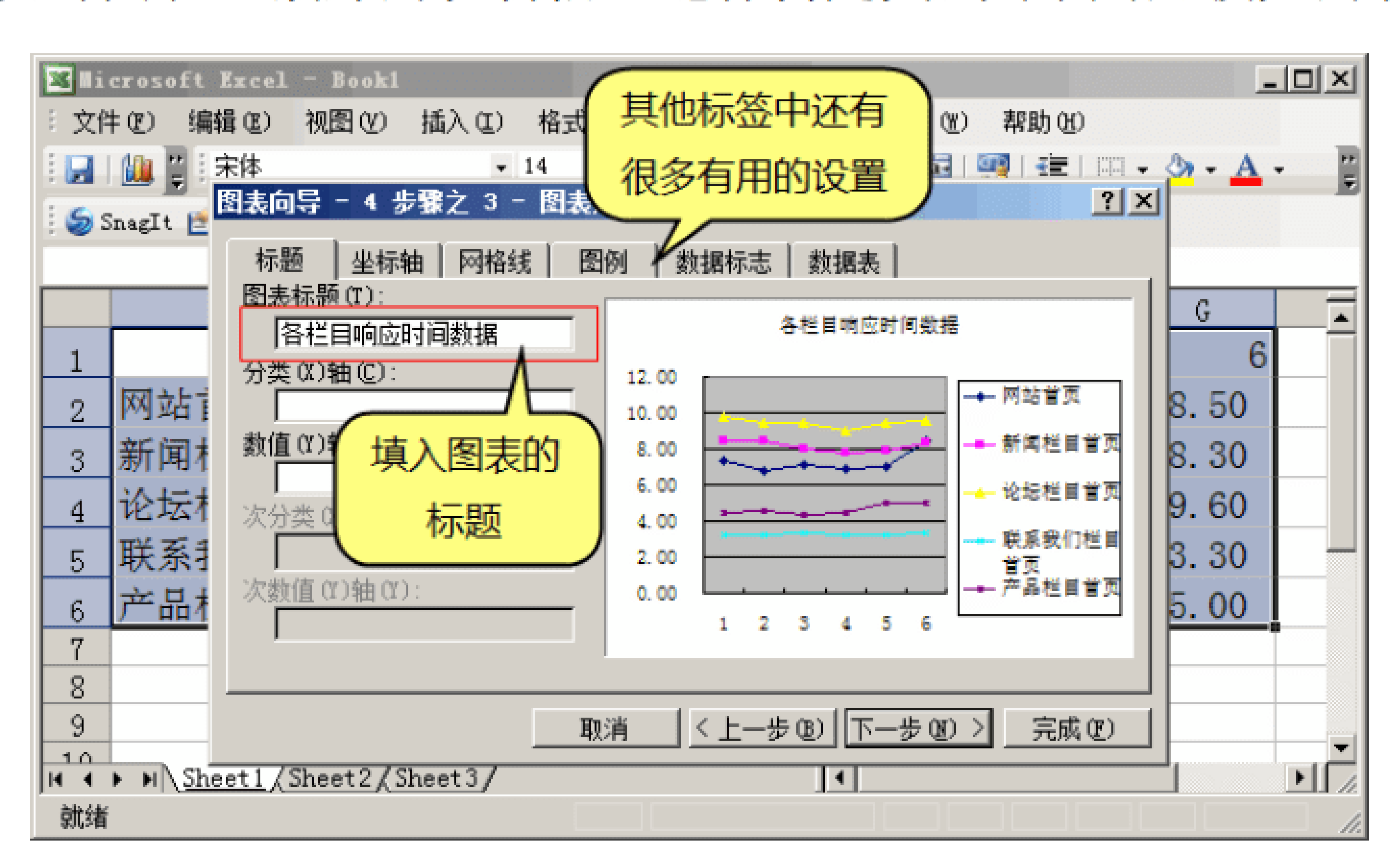


图 14-12　图表选项的设置

- 标题标签中的图表标题与 X、Y 轴标题，可以为整个图表起名字，并且对 X、Y 轴的数据进行说明、列出数值单位。
- 网格线标签中的显示网格线功能，使图表中各数值有公共参照物，利于比较。

- 选中数据标志标签中的各选项，可以使图表中线上各点附带上数值和说明。
- 数据表中显示数据表选项，可以将源表与由此生成的图表显示在一起。

依据实际的需要可以在这个步骤进行设置。感兴趣的读者可以亲自试验。

继续单击“下一步”按钮，将进入图表向导的最后一个步骤：图表位置的设置。它规定了图表是单独存在于一张 Excel 的新工作表，还是与源数据表格同处于一个工作表。我们在这里保持默认选择，即存在于当前工作表之中，如图 14-13 所示。

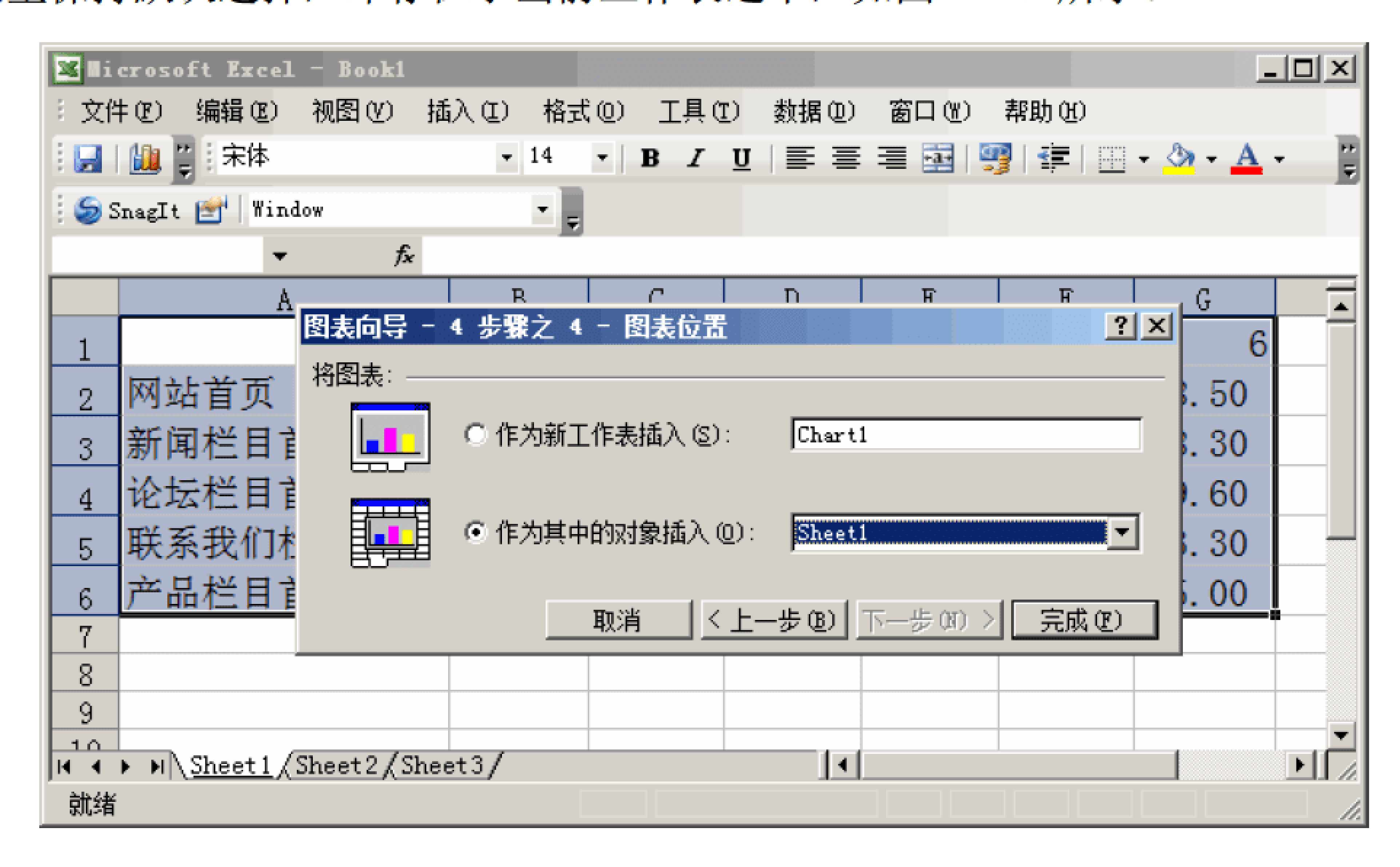


图 14-13　设置图表的插入位置

单击“完成”按钮，将完成图表向导的设置。生成的图表如图 14-14 所示。如果打算更改图表中各说明文字的大小与字体，可以双击图片空白处，以打开“图例格式”对话框进行更改，如图 14-15 所示。

需要说明的是，本书所举出的这个例子是以 Office 2003 中的 Excel 为例的，在 Office 2007 的 Excel 表中，界面基本类似，就不再赘述了。

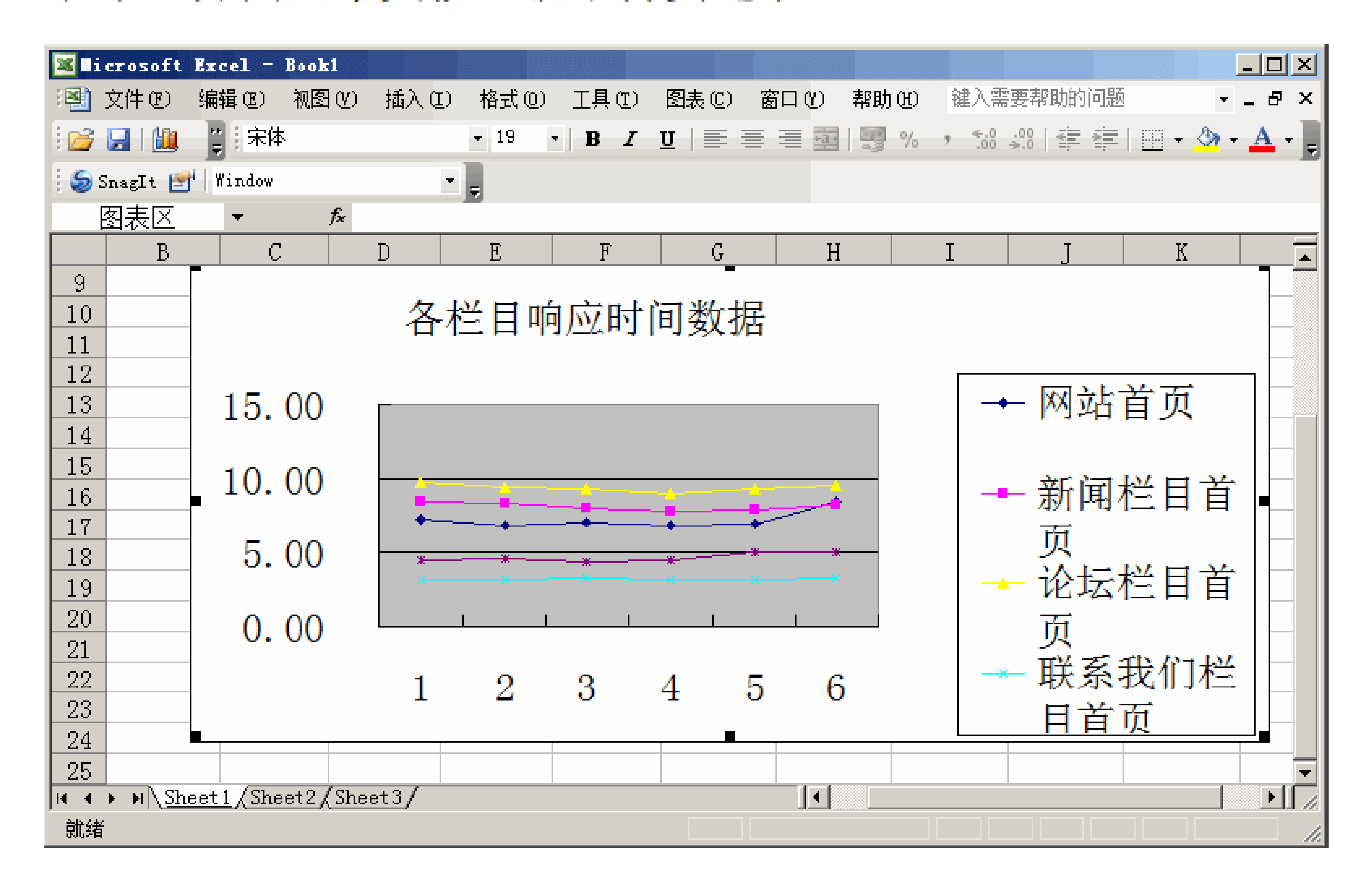


图 14-14　最终生成的折线图效果

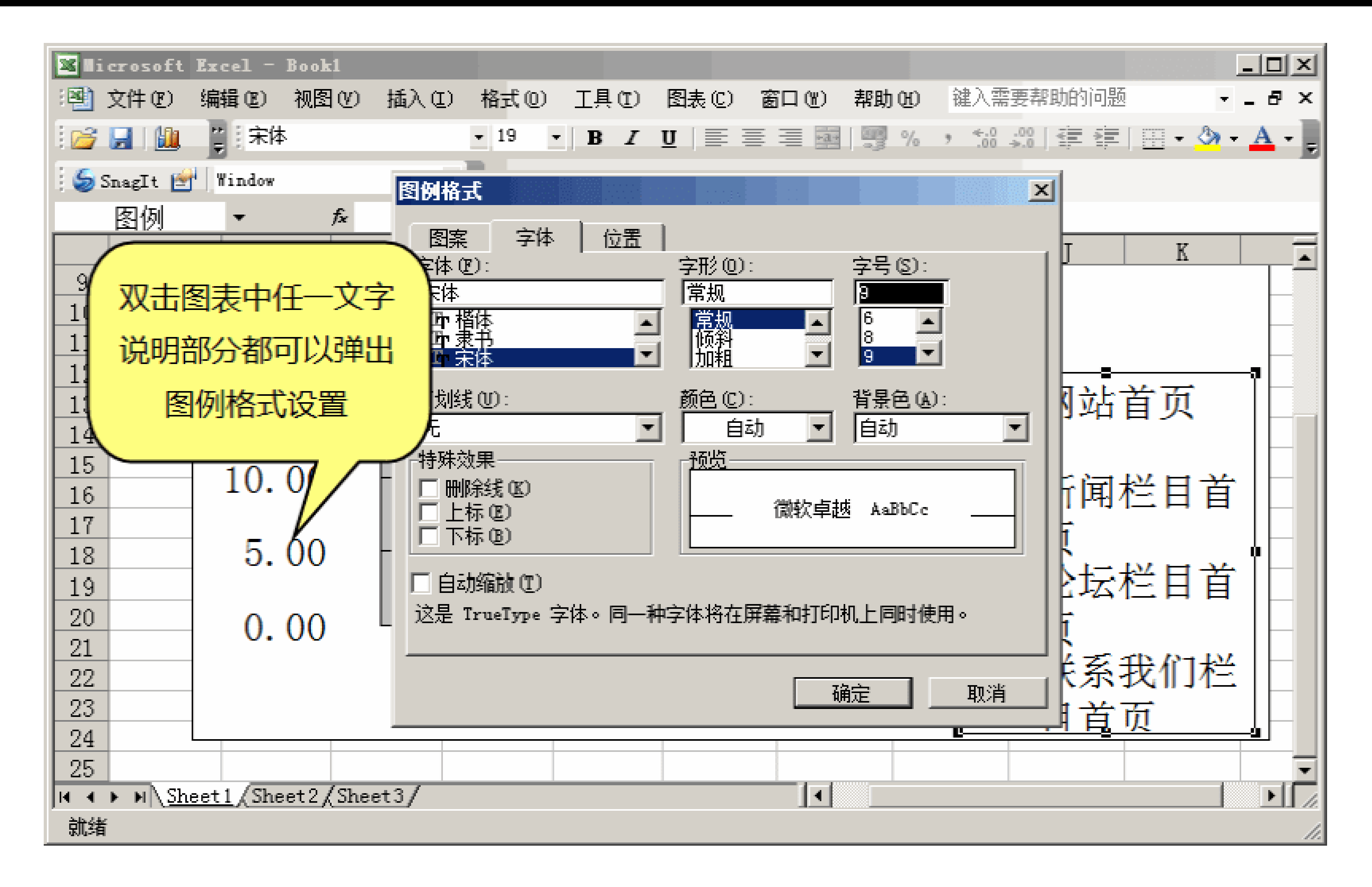


图 14-15　对图表中的文字设置图例格式

14.4　本 章 小 结

在本章中，我们学习了编写一份好的测试结果报告需要注意哪些问题。它们分别是：

- 首先要能判断测试数据的准确性。根据本章介绍的几种经验规则对数据进行筛选。
- 预先估计性能测试数据的大致状况。有了大致性能好坏的估计，对于报告结论就会有较清晰的认识。性能的大致信息可以通过本章介绍的一些统计学小知识来获得。
- 对性能测试报告进行细致的分析。
- 讲究技巧、注重逻辑性和易读性，通过文字、数据与图表的有机结合，编写测试报告。
- 与测试经理和其他相关人员进行沟通，最终完成测试报告。

通过以上这几个步骤，结合工作中的实践经验，相信读者一定能够完成一份不错的性能测试报告，为本轮性能测试的最后步骤，性能优化过程打下坚实的基础。

第 15 章　更多的性能测试工具

截至第 14 章，读者已经和小白一起将利用 LoadRunner 进行性能测试的全过程学习了一遍。但是，如果出于某种原因，在工作中无法使用 LoadRunner，那前面章节所介绍的知识就没有用武之地了吗？

回答自然是否定的。在本章的开头，有必要重提一下第 8 章所列出的测试流程图，如图 15-1 所示。



图 15-1　性能测试的主要过程

绝大多数性能测试都遵循这样的流程。可以看出，LoadRunner 的脚本（虚拟用户生成器）、场景（控制器）、测试结果（分析器）这 3 大部分只是测试过程设计的具体实现、执行性能测试、分析性能测试等少数几个步骤的具体实现。如果换用其他的性能测试工具软件，初学者也只需重点熟悉它的测试原理、测试如何执行等可能与 LoadRunner 不同的部分即可。

本章将对现有主流性能测试工具进行简单的概括式讲解，并对较方便获得的微软两种轻量级测试工具 WAS 和 VS Load Profiler 进行介绍，对于其他的工具，感兴趣的读者可以通过试用、查看帮助文档等尽快熟悉起来。对于学习一个新工具来说，多使用多看帮助文档可以说是一个捷径。

15.1　更多性能测试工具简介

本书的前面几章重点介绍了 LoadRunner 这一强大的性能测试工具，但实际上，在业内同时也使用着其他一些软件，本节将主要介绍 3 种分类。

15.1.1　性能测试工具的分类

根据应用的不同，性能测试工具可以分为 3 类。

（1）轻量级的性能测试工具，主要用于进行性能测试的某一个子分类（比如专门测试 Web 应用的 Web Server Stress Tool、专门测试网络性能的 PRTG 等），起到了总体评估和测试辅助作用。

（2）企业级性能测试工具，类似 LoadRunner、Rational Performance 这样，支持较多的协议，有强大的脚本函数、开发接口供测试工程师自定义开发测试脚本，并且能与其他测试软件相互共享数据的软件。

（3）白盒性能测试工具，能够提供行级代码分析，并具备指出对性能有较大影响的代码段等功能，例如本章 15.3 节要介绍的 Visual Studio 2008 分析器。

15.1.2　企业级性能测试工具简介

在企业级性能测试工具分类当中，除了本书讲解的 LoadRunner 之外，还比如有 JMeter 和 Webload 这两种工具。在本节的开始，也简要地进行一下介绍：

1．功能与性能测试工具JMeter

JMeter 是 Apache 组织的开放源代码项目，用 Java 实现，可以用来进行功能与性能测试。JMeter 进行测试前主要需利用图形化界面设置测试计划。该计划描述了执行测试过程中 JMeter 的执行过程和步骤，包括一个或者多个线程组（Thread Groups）用于设置负载信息（类似 LoadRunner 中的虚拟用户组）、逻辑控制（Logic Controller）、取样生成控制器（Sample Generating Controllers）、侦听器（Listener）用于获得性能计数器、定时器（Timer）、断言（Assertions）和配置元素（Config Elements）等。

JMeter 在测试 Web Service 方面应用很广。

2．性能测试和分析工具Webload

Webload 是 RadView 公司推出的性能测试、分析工具，它能够通过模拟真实用户的操作，生成压力负载来测试 Web 的性能，而且还可以自动执行。它的测试过程主要通过设置日程（Agenda）来实现。

总体来说，Webload 的特点有两个：

- 可以测试 Adobe 的 Flex 应用，这在其他的 Web 应用性能测试工具中是较少见的（本书介绍的 LoadRunner 是支持的）。

- 测试脚本用 JavaScript 编写，符合 Web 开发人员的习惯，学习成本低。

【企业测试工具的共同特点】

对于企业级性能测试工具，虽然种类很多，名词表述、操作表现形式各异，但是它们中的绝大多数都包含有负荷的创建（多数通过设置虚拟用户的方式）、录制或者编写模拟用户操作的测试脚本、获取性能计数器以及结果分析这几个模块。

15.1.3　轻量级测试工具的优点

除了以上所介绍的企业级测试工具，有的时候在工作中由于时间关系，测试工程师只打算了解当前 Web 应用基本的性能状况，那么轻量级的性能测试工具就会很有用。虽然它们的功能无法和诸多专业工具相比，但也具备如下的特点：安装体积小、使用方便、分析较快速直观，而且一般是免费的。

在本章的第 15.2 节，笔者将介绍微软出品的轻量级性能测试工具：WAS（Web Application Stress Tool，简称 WAS）。针对白盒性能测试分类，本章最后一节将以 Visual Studio 2008 分析器（Analyer）为例进行讲解。另外，在本书的附录 A 中列出了更多性能测试工具以及下载的网址，供感兴趣的读者参考。

15.2　WAS 的使用简介

微软的 WAS 是一个很小的工具，而且它的存在已经有 7 年的历史了。WAS 能够通过脚本模拟 100 个并发虚拟用户对 Web 应用的访问，在脚本中也可以模拟实际用户的一些点击操作，从而获得 Web 应用的性能测试结果，为进一步分析提供数据。WAS 同样能够连接远程 Windows 网站服务器的性能计数器（Performance Counter），通过对其的获取，来找到 Web 应用运行过程中可能出现的瓶颈。

15.2.1　WAS 的安装与启动

WAS 的安装地址在微软官方网站上搜索 Web Application Stress Tool 即可得到，是一个名为 Setup.exe 的安装包。它的安装也很简单，运行下载的 Setup 程序，保持默认选项，单击多次“下一步”按钮即可完成。

【需要 VC5 运行库的问题】

由于 WAS 开发实践较早，其运行必须的 VC5 运行库文件（msvcp50.dll）在较新的操作系统比如 Vista 中已经没有了，可以采取的解决方法有两种：在 Windows 2000、Windows XP 或者 Windows 2003（最好是 Windows 2000）的机器上安装此工具；或者，将 msvcp50.dll 文件复制到 WAS 的默认安装目录，再重新运行安装文件 Setup.exe 即可。

安装完毕后，WAS 可以通过在桌面单击“开始”|“所有程序”|“微软 Web Application Stress Tool|微软 WAS 工具”命令来启动。

WAS 与 LoadRunner 进行性能测试的过程类似但要简单直接得多，整个测试过程可以分为录制脚本、执行脚本和分析结果 3 大部分。在下面的几节中我们将对这些步骤进行

讲解。

15.2.2　录制脚本

当 WAS 工具启动时，界面如图 15-2 所示。

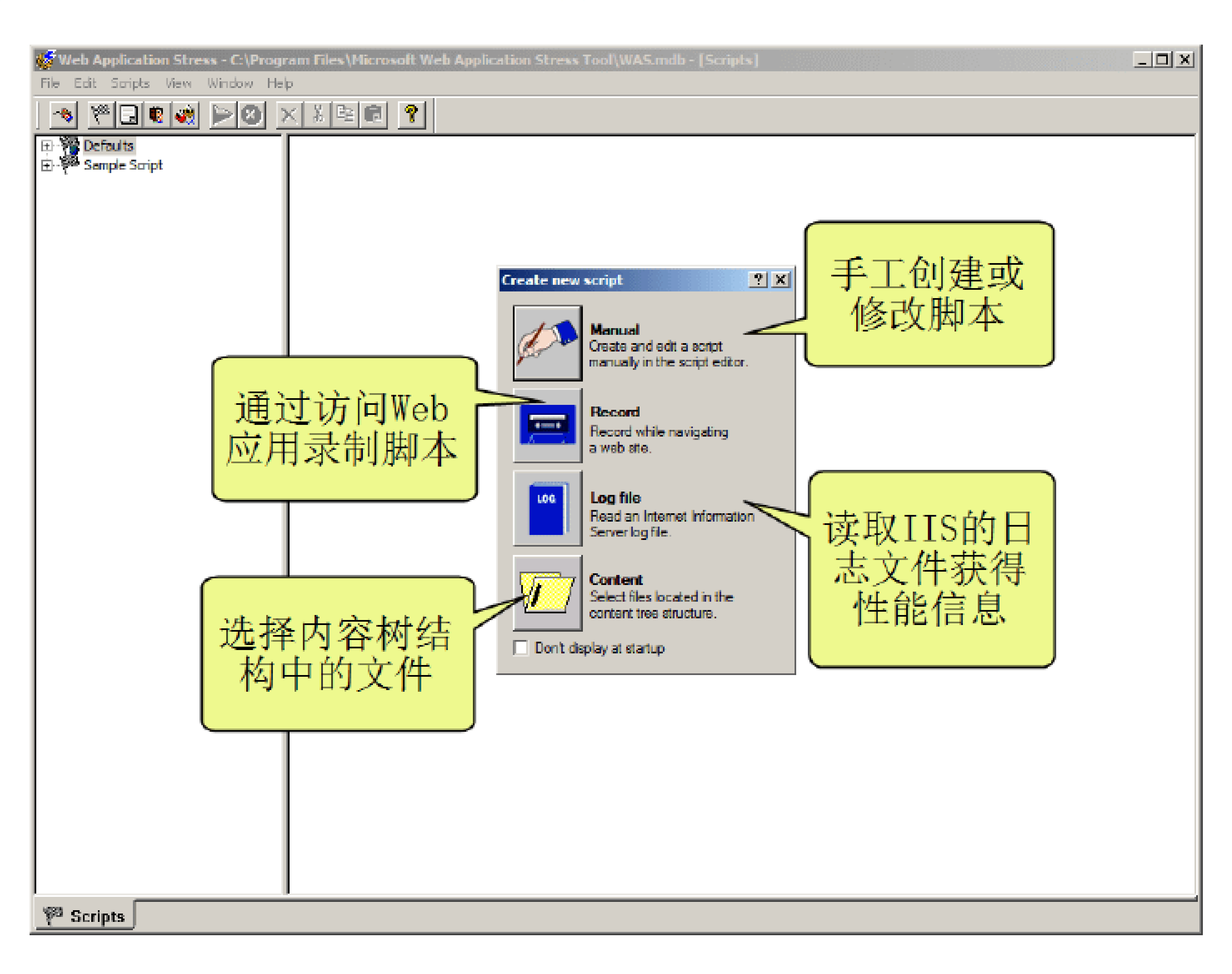


图 15-2　WAS 的启动界面

从图 15-2 中可以发现，创建新脚本有 4 种选择，分别是：

- 在脚本编辑器中手工创建或者修改脚本（Manual）。
- 通过访问 Web 应用录制脚本（Record）。
- 读取 IIS 日志文件（Log file）。
- 选择内容树结构中的文件（Content）。

由于篇幅所限，本节只以选择录制脚本这个选项为例进行介绍。

（1）在图 15-2 中单击 Record（录制）按钮后，WAS 界面将变为 Browser Recorder-Step 1 of 2（浏览器记录）设置对话框，其中有多个复选框，选中后分别可以设置“录制请求之间的延迟”（类似思考时间）、“录制浏览器 Cookie”和“录制主机头”（Host header），如图 15-3 所示。默认情况下，这 3 个复选框都是可以不用选择的。



图 15-3　录制属性设置

（2）单击 Next 按钮，即可完成录制设置。此时，WAS 将弹出一个默认浏览器窗口（为了保证测试效果，之前最好把所有浏览器窗口都关闭），在其中的地址栏中输入被测试

Web 应用的 URL 并回车之后，WAS 界面如图 15-4 所示。

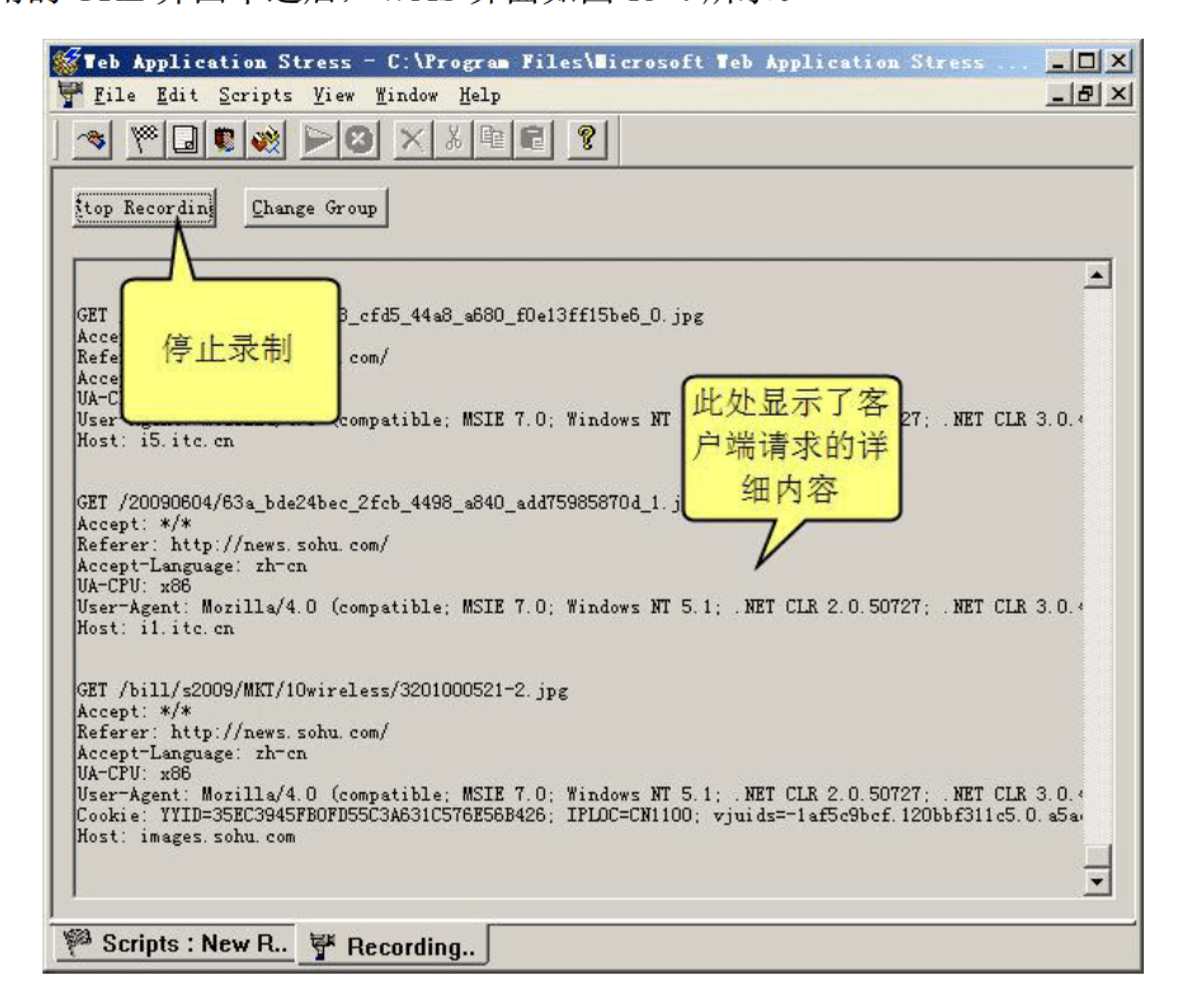


图 15-4　使用 WAS 录制页面操作脚本

此时，录制已经开始，测试工程师需要按照要求在真实的网页上执行各种操作直到单击 Stop Recording（停止录制）按钮结束脚本的录制。

15.2.3　执行测试

单击结束录制按钮之后，WAS 将返回脚本编辑界面，并生成名字为新录制脚本的节点，如图 15-5 所示。读者也可以将这个名字修改为更有意义的文字。

展开图 15-5 中新录制脚本节点，可以看到若干属性子节点，分别对录制脚本的回放参数做出了设置。实际上，图 15-5 右边内容视图显示的就是选择 Settings（设置）叶子节点后的结果。

【Perf Counters 的问题】

在图 15-5 左侧“新建录制脚本”（New Recorded Script）节点下的诸多选项中，如果 WAS 不是安装在 Windows 2000 上，则 Perf Counters 设置将由于无法连接到服务器（哪怕是本机）而无法添加。若在其他系统上运行，则可以用前面章节所介绍的各种性能计数器获取方法（比如 typeperf 程序）取得数据。好在 WAS 本身是个简单的工具，它的优点在于能方便快捷地得到有关 Web 应用的大致性能信息，即使有些设置不很通用，还是可以利用别的工具替代而获得，因此，实际工作中还是有不少人在使用它进行性能的粗略评估。

WAS 还允许测试工程师使用多个客户端机器测试你的网站，这可以通过 Client 属性节点来设置。当一次测试开始时，WAS 会自动地与所有设置好的客户机（Client）取得联系，

向它们传输所有的测试信息（包括测试脚本项，页面组和用户定义信息），启动和停止它们的测试，然后收集测试结果。

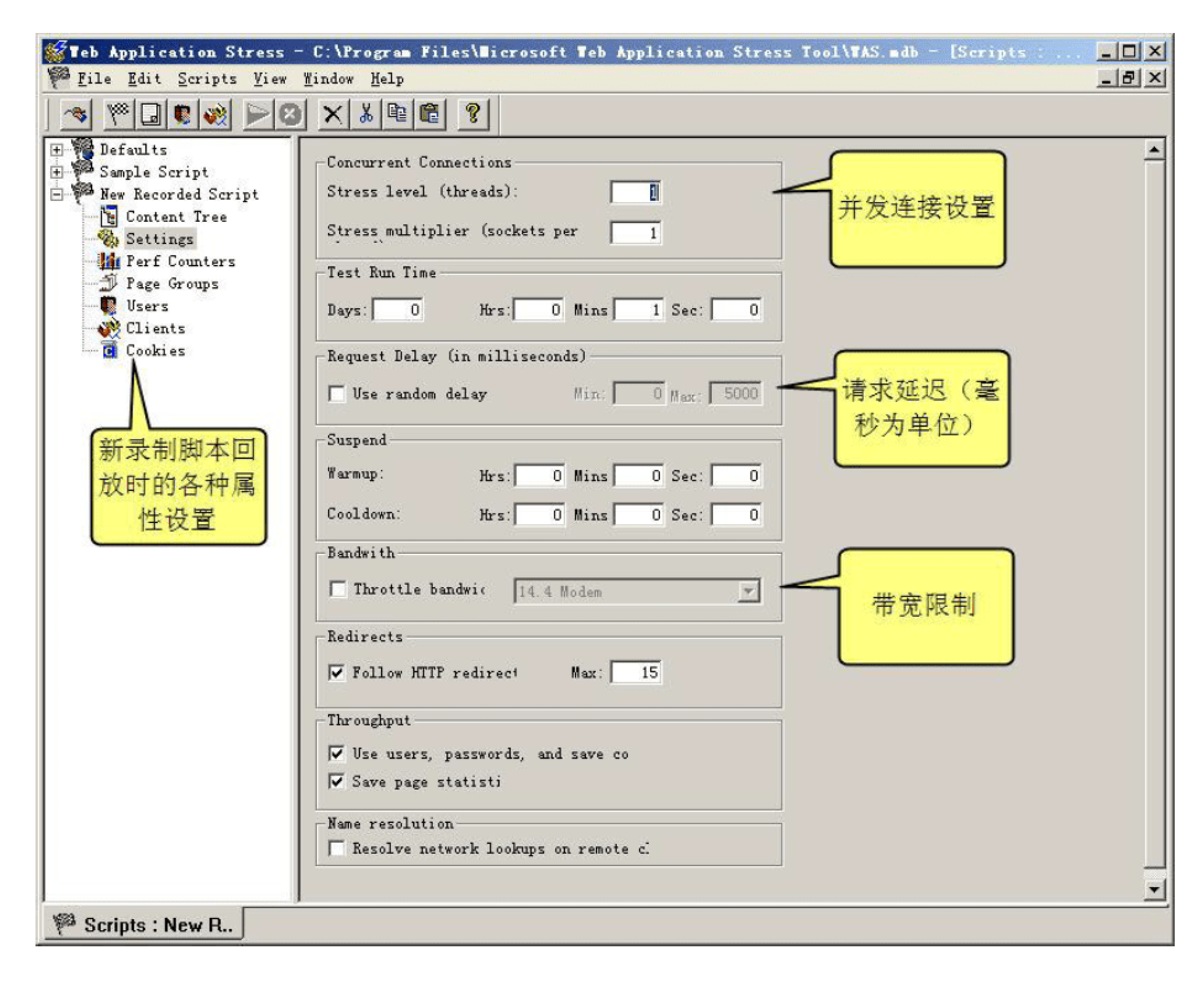


图 15-5　录制脚本完毕后对脚本的运行参数进行设置

另外值得一提的是 Users（用户）叶节点，单击该节点后，右边的内容视图将出现用户组的列表，WAS 默认设置的一个用户组（Default）列于其中。双击该默认组，则会出现其中每一个用户的名称和密码设置（系统默认数量为 200 个），如图 15-6 所示。

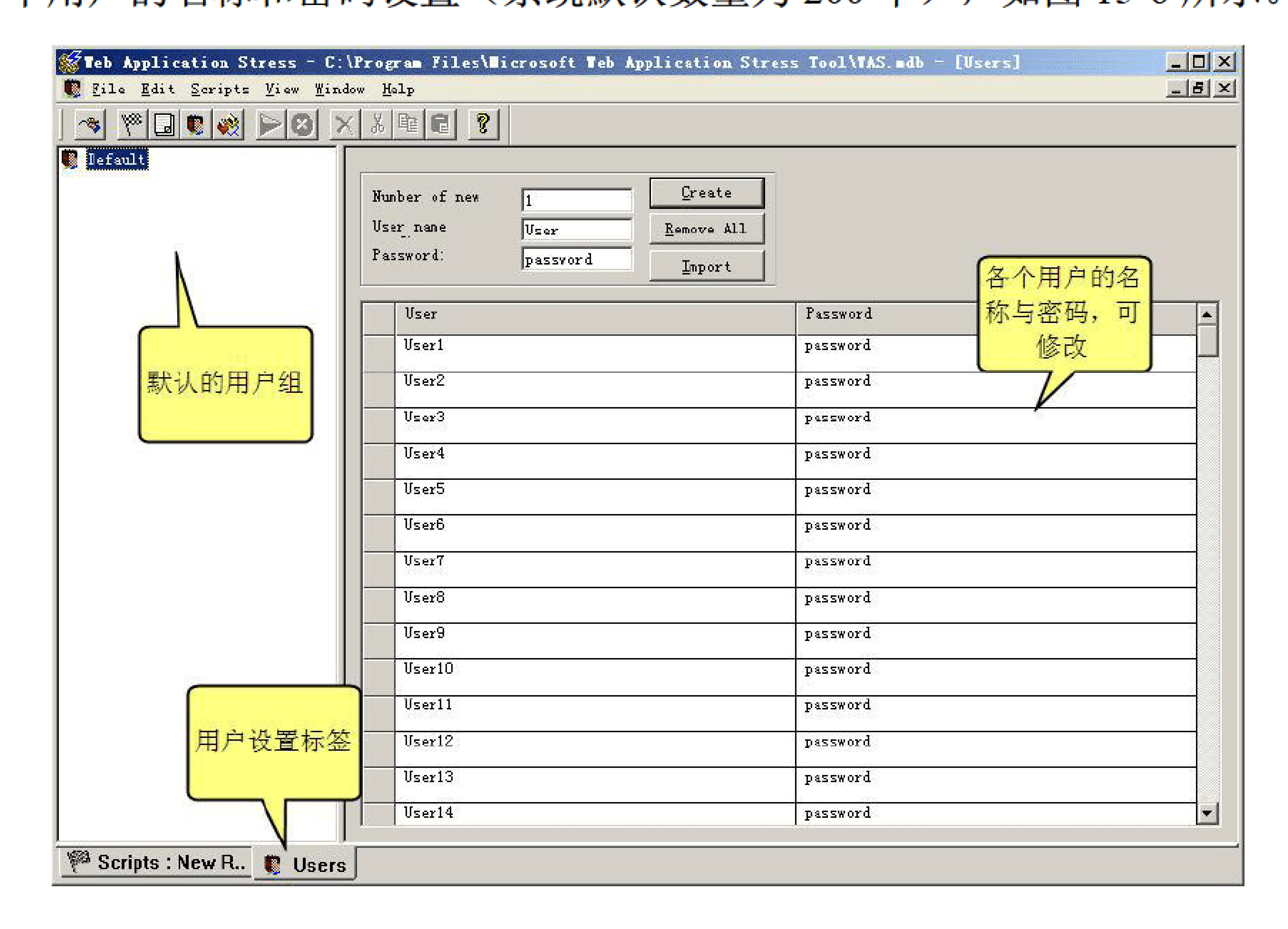


图 15-6　运行脚本前的用户设置

依次对新录制脚本下的各属性节点进行设置，待一切确定后，单击 WAS 工具栏中的 Run Script（运行脚本）图标按钮，就可以回放刚才录制的脚本，获得性能数据了，如图 15-7 所示。

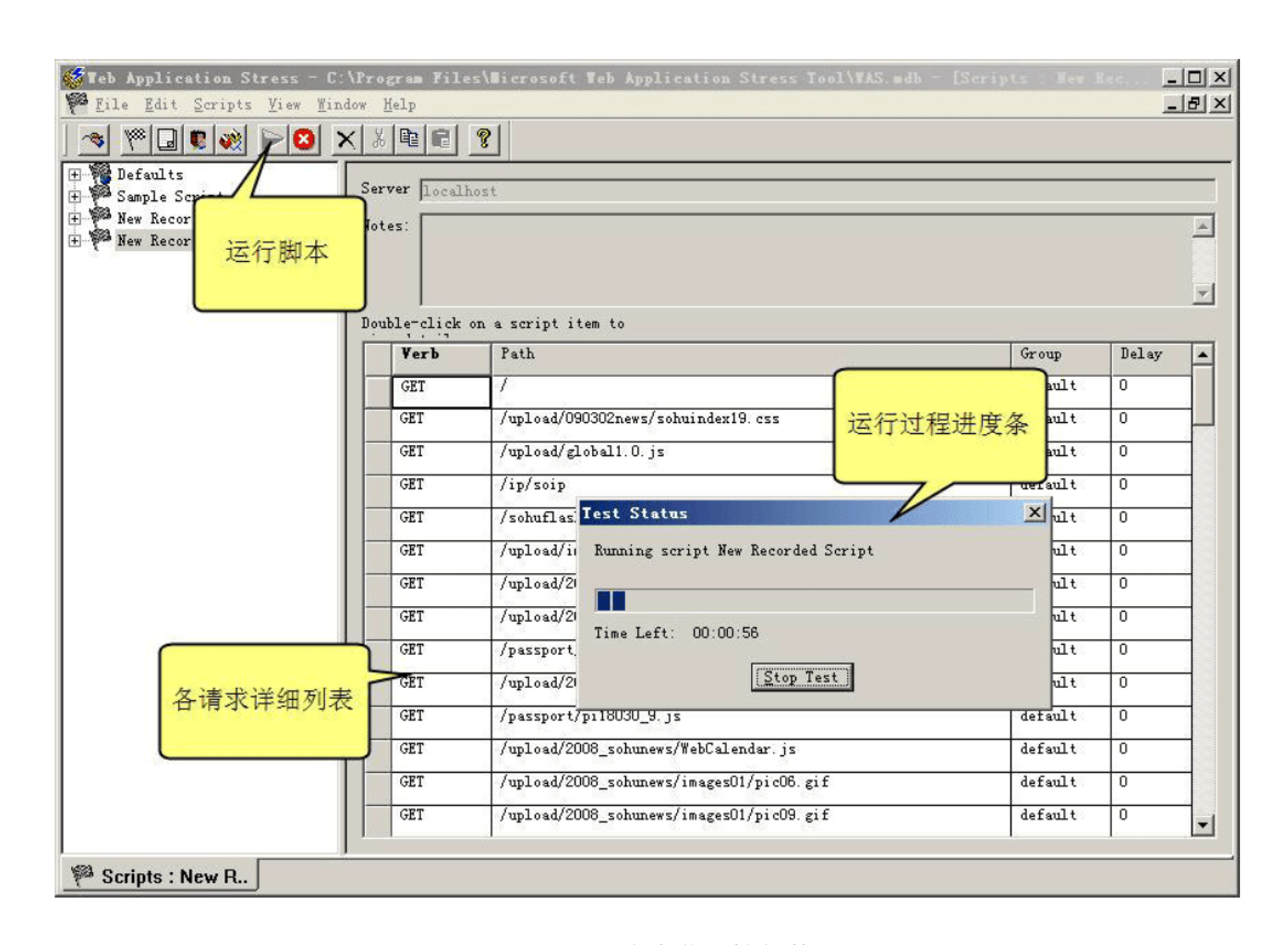


图 15-7　运行脚本获取性能数据

15.2.4　分析结果

当脚本运行结束或测试工程师单击停止运行图标之后，测试工程师进而可以通过选择 WAS 界面下方的 Report（报告）选项来显示此次运行的结果，如图 15-8 所示。

WAS 的测试结果可以提供 HTTP 返回码、点击次数、请求数量以及统计、错误数量等多项信息。从整个过程来看，WAS 具备一些类似 LoadRunner 的特点。

随着微软开发工具的不断进步，后来又推出了 WAS 的升级版：应用程序中心测试组件即 ACT，使用方法与界面还是比较简单的，其可以直接从 Visual Studio 中启动，感兴趣的读者可以熟悉一下。

如今，在工作中广泛使用的主流 Visual Studio 版本已经演变为 2005 或者 2008，它们的“团队开发套件”（Team Suite）版本均包含一个用于测试性能的组件，称为分析器（Analyzer）。有了它，性能测试与开发代码的结合越来越紧密了。在 15.3 节将简要介绍 Visual Studio 2008 中分析器的使用方法。

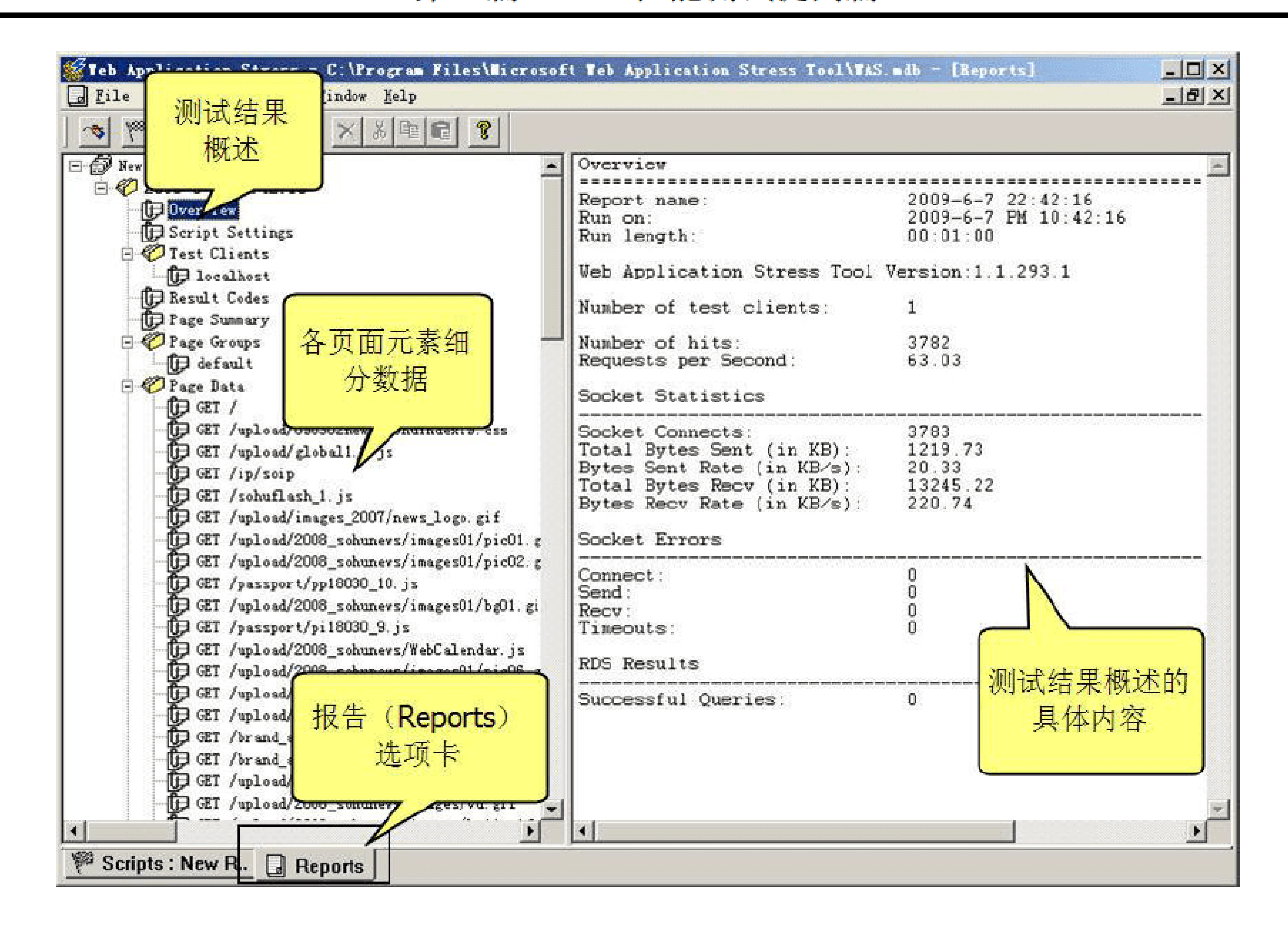


图 15-8　测试结果界面

15.3　Visual Studio 2008 中的性能测试工具简介

前文提到，微软著名的开发工具 Visual Studio 2008 团队系统开发版（Team System Development Edition），或团队套件（Team Suite）内部就包含了一个性能测试工具——分析器（Analyzer），它可以很方便地对处于开发过程中的 Web 应用进行测试。当然，它更可以对已经上线的网站进行测试，不过，由于该工具处于 Visual Studio 集成开发环境内，更多情况下需要被测试 Web 应用源代码的支持，因此使其具备更多的白盒测试特性。

Visual Studio 2008 中的分析器是默认安装的，如果需要自定义安装，请在安装界面中选择性能工具，如图 15-9 所示。

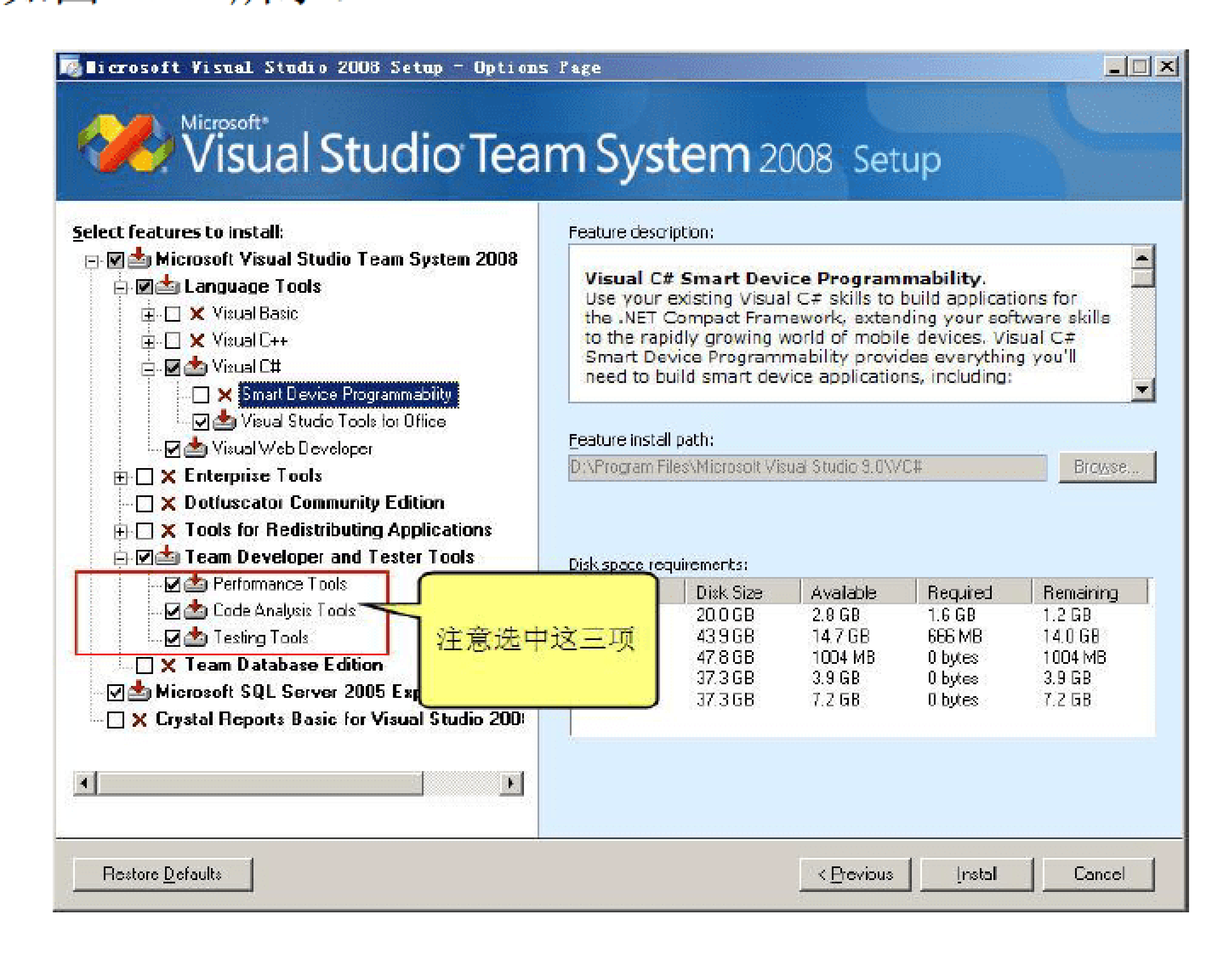


图 15-9　Visual Studio 2008 安装时有关是否安装性能工具的选择设置

本书并不会在这里详细讲解 Visual Studio 中的分析器，只为开拓读者的眼界，展现性能测试的另一种方法。感兴趣的读者可以选择相关书籍进行深入阅读。

15.3.1　性能测试流程

分析器的性能测试流程与 LoadRunner 等有所不同，它可以分为：

- 通过性能向导创建分析会话并设置分析方法（采样法或者检测法）。
- 在浏览器中启动被测试 Web 应用，并在分析器的性能浏览器中设置分析标记。
- 结束测试后浏览性能报告摘要。
- 通过调用树、热路径等功能找到性能问题所在。

从以上步骤可以看出，分析器对测试工程师的开发背景有所要求。

（1）分析器（Analyzer）可以从 Visual Studio 2008 中的分析（Analyze）菜单中启动，该菜单下有如下几个菜单项，如图 15-10 所示。单击“打开性能向导”（Launch Performance Wizard）菜单项，性能向导将被运行。首先，读者需要选择被测试软件的类型，如图 15-11 所示。由于本书的目标在于测试 Web 应用程序，因此在图 15-11 的下拉列表框中选择 ASP.NET 即可。从这个下拉列表框中也可以看出，分析器是有支持平台方面的局限性的。

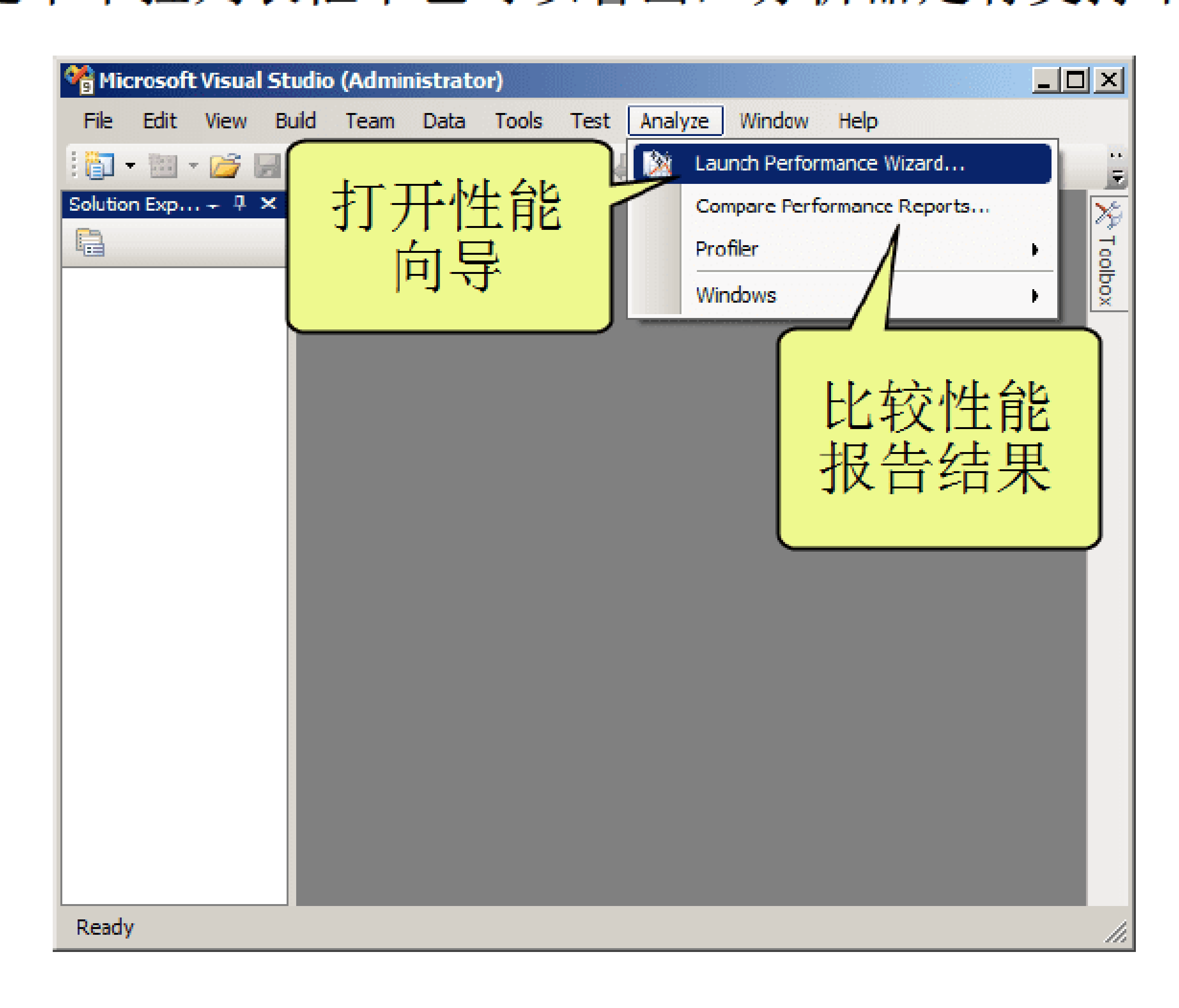


图 15-10　Visual Studio 2008 中的分析器

【若被测试项目已经打开】

如果 Visual Studio 2008 中已经有了打开的 ASP.NET 应用项目，那么这个下拉列表框将默认选中当前项目，不必在 EXE、DLL 和 ASP.NET 之间进行选择了。

（2）继续前文的叙述，单击 Next 按钮，在测试网址文本框中输入被测试 Web 应用的 URL（按照实际情况输入即可），如图 15-12 所示。

（3）单击 Next 按钮，向导进入第 3 步：对分析方法的选择，如图 15-13 所示。对于使用 ASP.NET 开发的 Web 应用，系统默认选中第 2 个单选按钮：Instrumentation（检测法）。

【采样与检测的区别】

分析器采用采样法时，其运行方式类似食品生产流水线上的抽检，定时中断程序以查看和收集当前程序运行信息，分析器不改变代码本身，这很类似本书中主要讲解的

LoadRunner 的运行方式。而检测法则不同，分析器会在代码的每个函数开头与结尾处注入“探针”，从而可以获得运行函数所花的时间等更详细的信息，因此，这可以称为一种白盒性能测试。由于检测法获取信息量较大，因此需要花费的分析时间会更长。



图 15-11　选择被测试软件类型



图 15-12　输入被测试的 Web 应用 URL

【Web 应用与检测法】

由于 Web 应用的特点：用户请求比较随机，运行过程中各环节影响因素也较多，因此对资源消耗具备缺乏普遍规律的特点。而采样法是每隔固定时间（默认为每 1000 万个 CPU 周期）对被测试软件进行一次信息收集工作的，因此，当 Web 应用等待磁盘、网络等资源时，采样法将无法获得信息。这就是分析器默认采用检测法测试 Web 应用性能的原因。

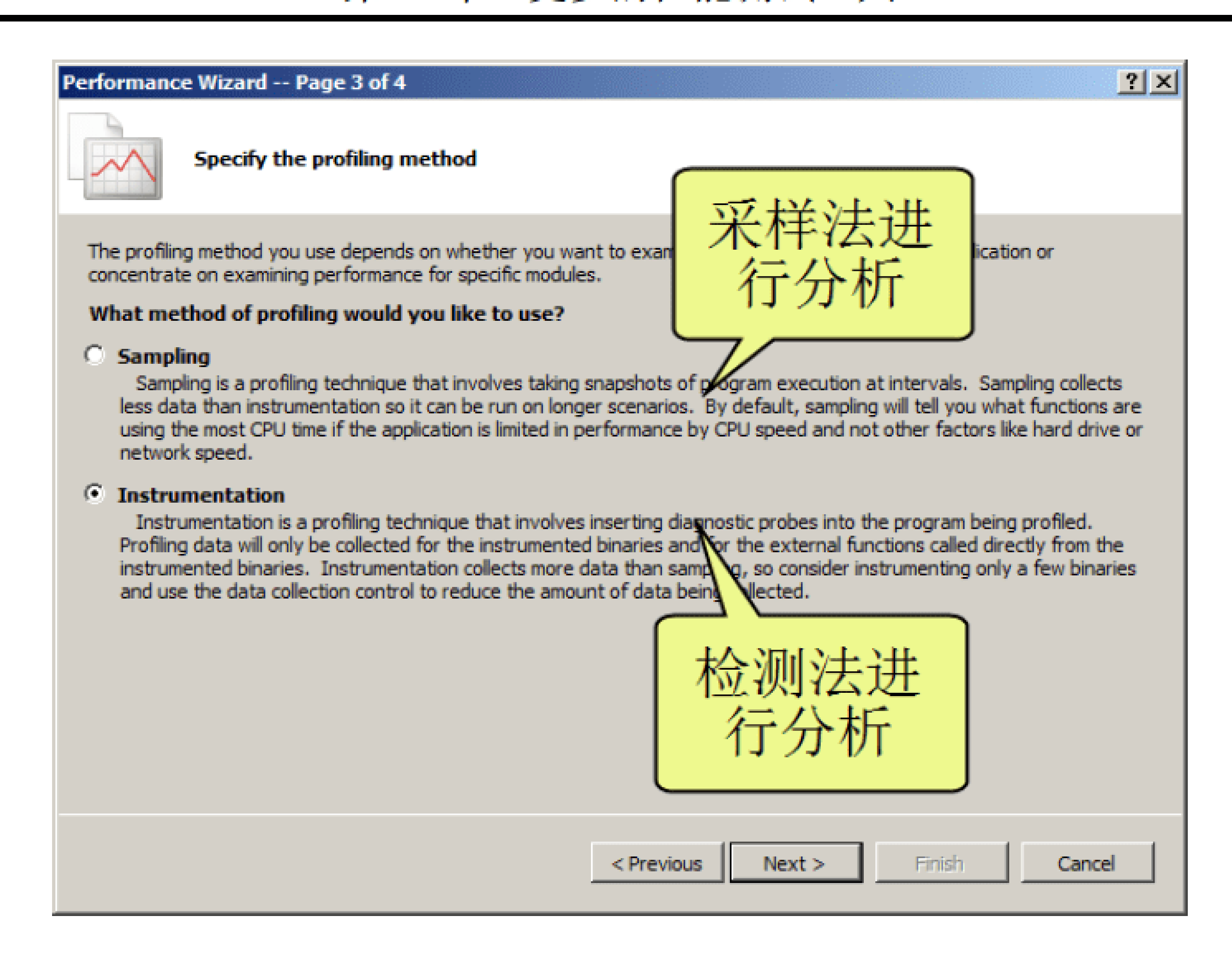


图 15-13　分析方法选择

（4）单击 Next 按钮，即可完成性能向导的设置。同时，Visual Studio 2008 的集成开发环境（IDE）界面也将变为如图 15-14 所示。

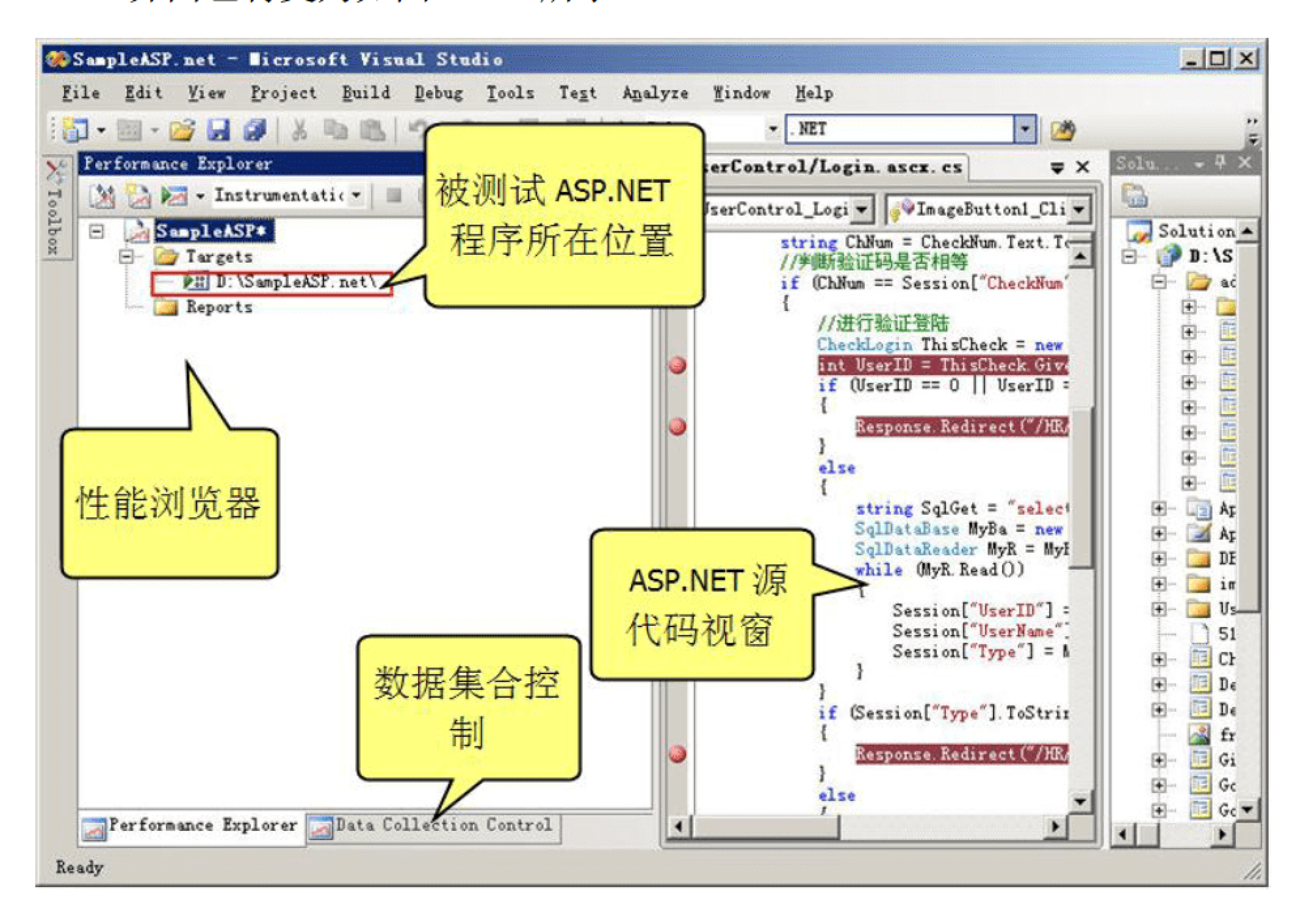


图 15-14　性能向导设置完成后性能浏览器的界面

【两个选项卡很有用处】

注意图 15-14 中增加的性能浏览器视图，它实际上包含两个选项：Performance Explorer（性能浏览器）与 Data Collection Control（数据集合控制）。在“性能浏览器”选项卡中，可以完成性能测试执行的开始、停止、分析方式的选择等功能。而 Data Collection Control 选项卡则非常类似于 LoadRunner 中虚拟用户的设置。

（5）选择 Performance Explorer 选项卡，单击工具条中的向右箭头图标，可打开包含 3

个菜单项的下拉列表框，如图 15-15 所示。

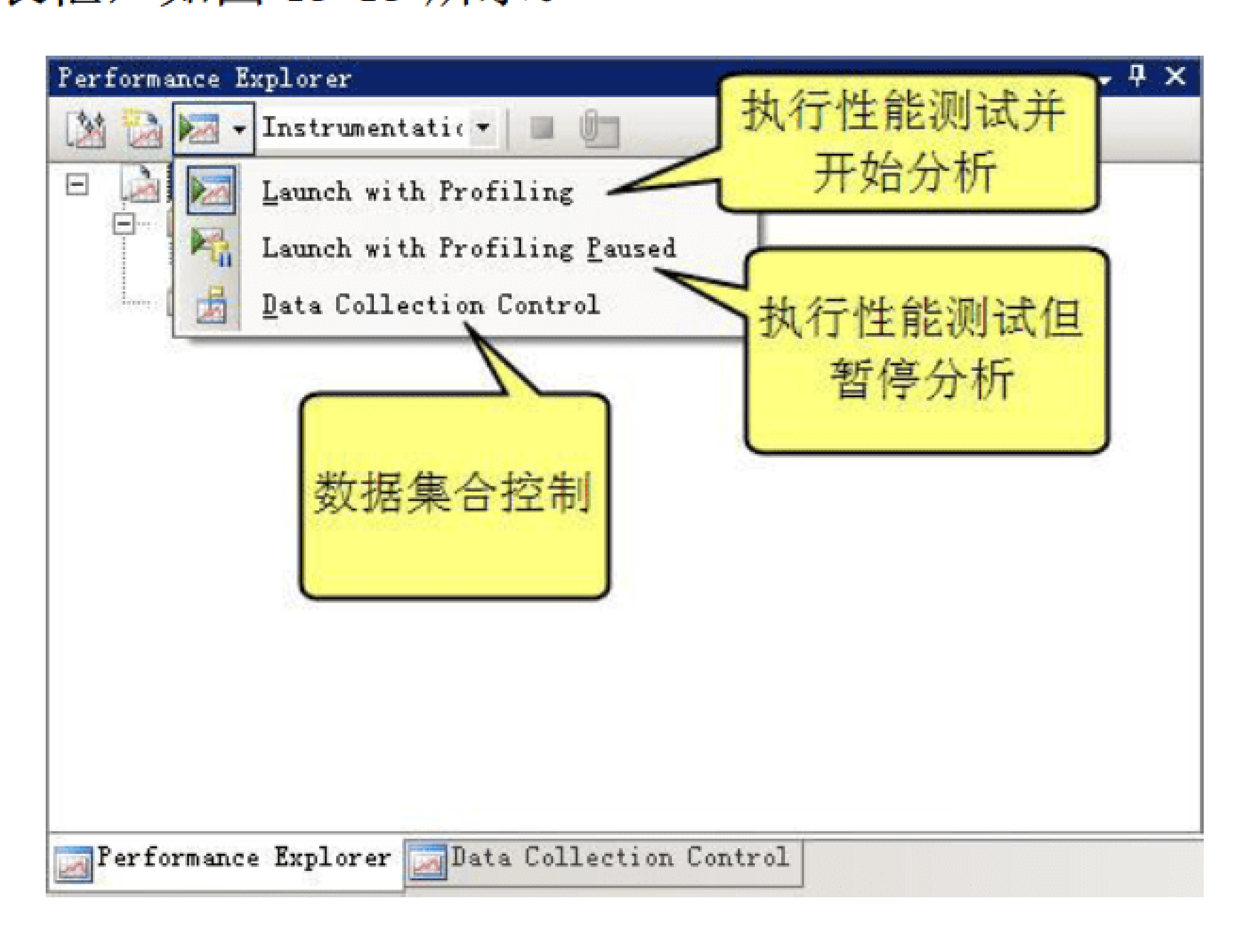


图 15-15　执行性能测试的选择菜单项

（6）测试工程师只需要选择第一个选项 Lauch with Profiling 执行性能测试并开始分析，整个 ASP.NET 就会在一个新的浏览器窗口中打开（同时还打开了 Visual Studio 2008 中自带的 Web 服务器以提供服务），可以在其上自由地操作直到关闭整个浏览器。

（7）将测试 ASP.NET 应用关闭之后，Visual Studio 2008 将生成一个后缀名为 vsp 的报告文件，其结果也会显示在界面上，如图 15-16 所示。

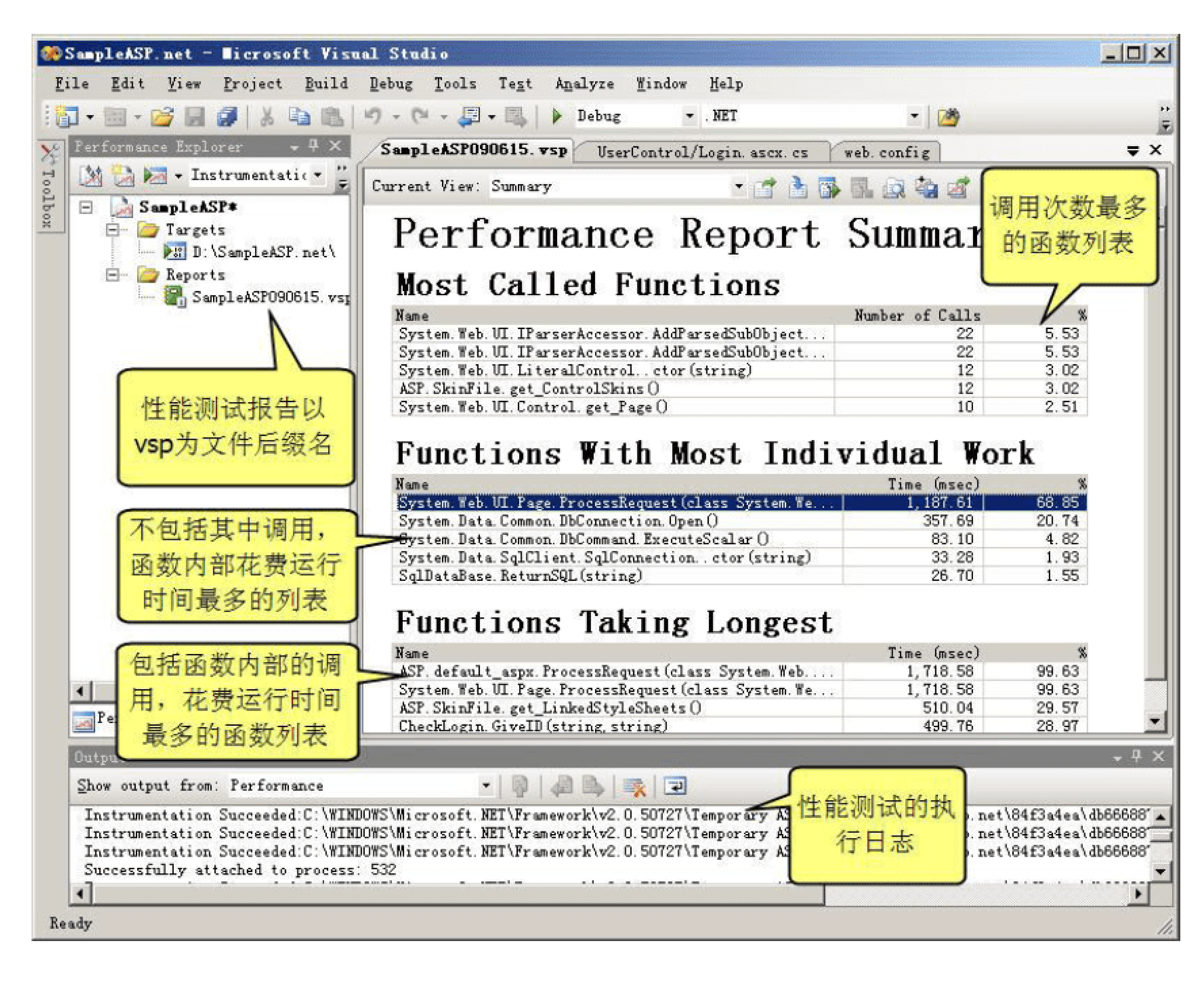


图 15-16　执行完毕后性能测试报告

在图 15-16 的性能测试报告中，列出了被调用次数最多的函数、代码内部（即执行本函数的代码，并不包括其中调用其他函数的代码）所花费运行时间最多的函数，以及总体花费运行时间最长的函数列表。

15.3.2　调用树与热路径

在进行性能测试结果的分析之前，有必要了解调用树（Call Tree）与热路径（Hot Path）的背景知识。

【调用树的含义】

所谓调用树，简单地说，就是含有当前正在运行函数的代码中被调用过程的所有模块。这些模块都是以当前运行模块为出发点或者“根”，逐层嵌套，最终形成了一棵多处分叉的类似“树木”。

而从树根到不同树梢的行进路线就形成了若干条路径。在很多条这样的路径之中，热路径是在调用树中最花费时间的那一条路径。因此，解决热路径花费的时间问题可以说是解决代码性能问题的关键之一。在 15.3.3 节将介绍热路径的查看方法。

15.3.3　测试实例

有了调用树与热路径的概念，通过分析器的运行结果发现代码中的性能瓶颈将变得很轻松。在图 15-16 中性能测试报告上方的 Current View（当前视图）下拉列表框中，选择 Call Tree（调用树）选项，如图 15-17 所示。调用树的结果则如图 15-18 所示。

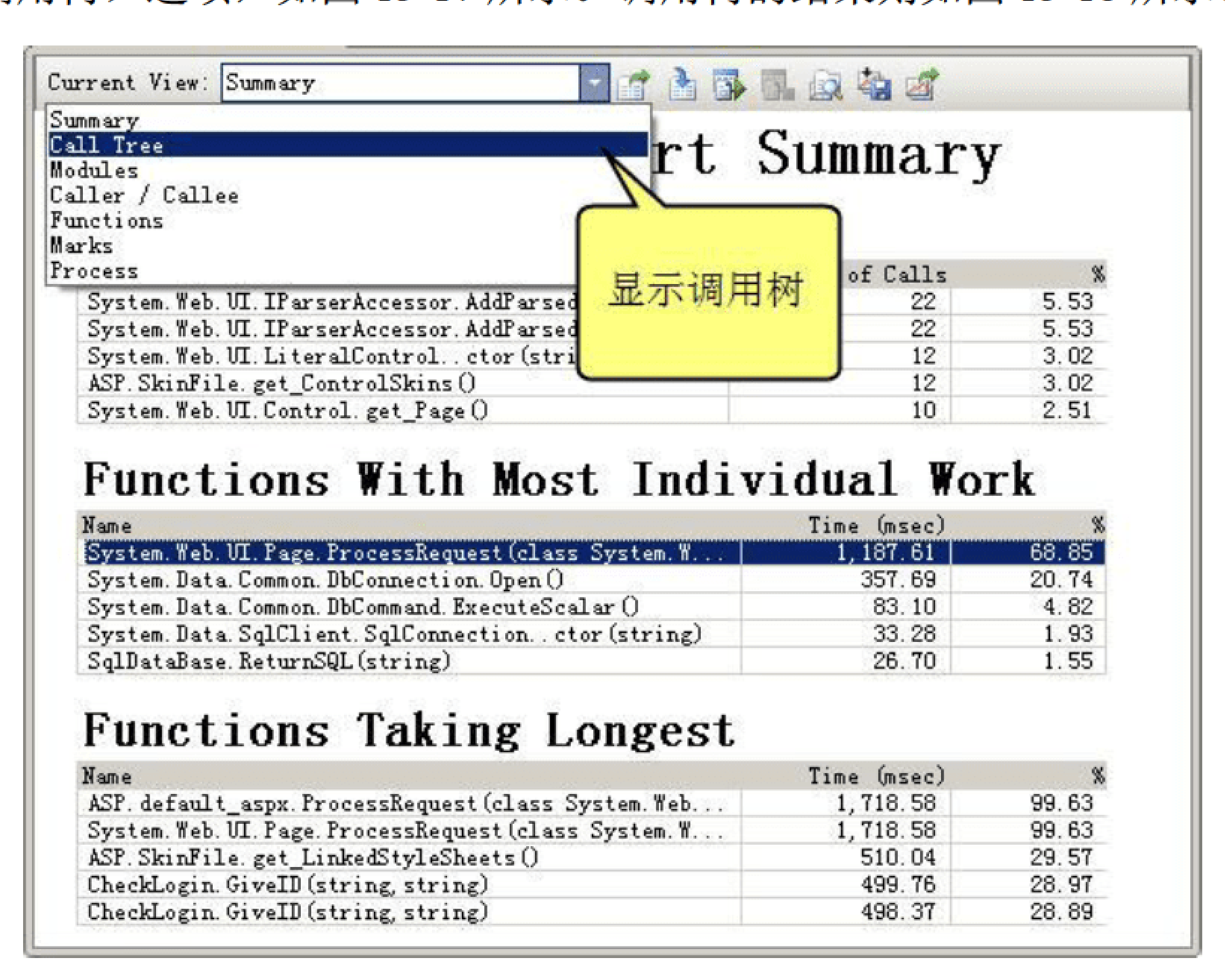


图 15-17　在当前视图中选择显示调用树菜单

在图 15-18 中特别标出的两列数据分别代表 Included Time（内含时间）和 Excluded Time（专有时间）。注意其中内含时间与专有时间的区别如下。

- 内含时间：包含函数中被调用的其他函数运行所花费的时间。
- 专有时间：不包含函数中被调用的其他函数运行时间，只是本函数代码运行的时间。

【实战演练】

内含时间与专有时间的区别可以通过下面一个简单的实际例子来说明。如代码 15-1 列出了一个简单的函数调用。

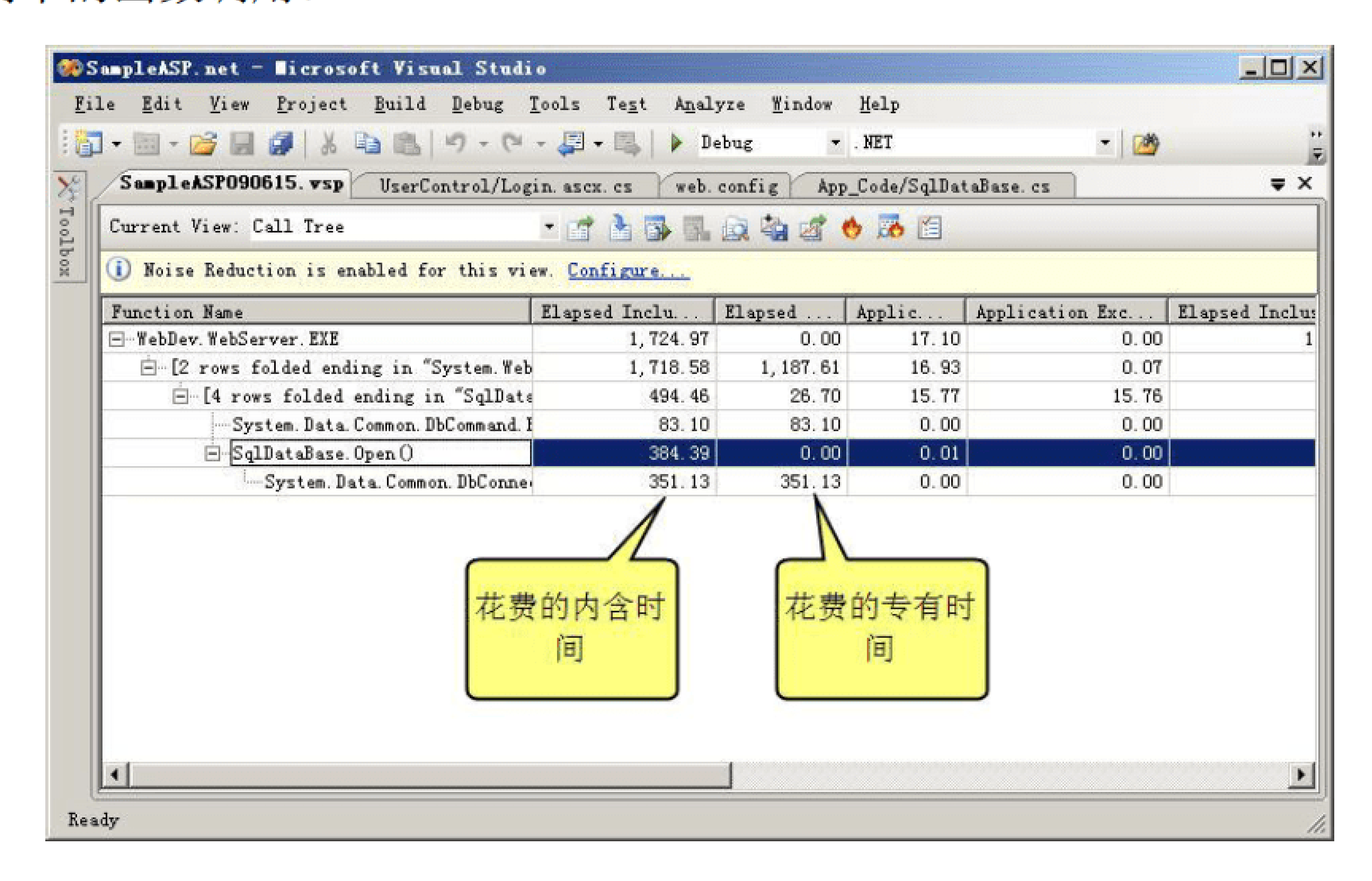


图 15-18　调用树的详细列表

代码 15-1　内含时间与专有时间的例子：一个简单的调用

```
int A() {
    // 直接返回
    return 0;
}

int B() {
    // 调用 A
    return A();
}

int main() {
    // 调用 B
    return B();
}
```

测试结果中的内含时间与专有时间数据如表 15-1 所示。

表 15-1　代码 15-1 的测试数据

函 数 名 称	内含时间（毫秒）	专有时间（毫秒）
Main	500	50
B	450	200
A	250	250

在表 15-1 中，A 函数没有再调用其他函数，因此内含时间与专有时间相等。B 函数调用了 A 函数，因此内含时间等于专有时间与执行 A 所花费的时间 250 毫秒之和。

在图 15-18 中，单击工具栏中、处于调用树列表右上方的火焰图标按钮，即可以发现当前代码中的热路径，即运行花费时间最长的一条函数调用路径，如图 15-19 所示。

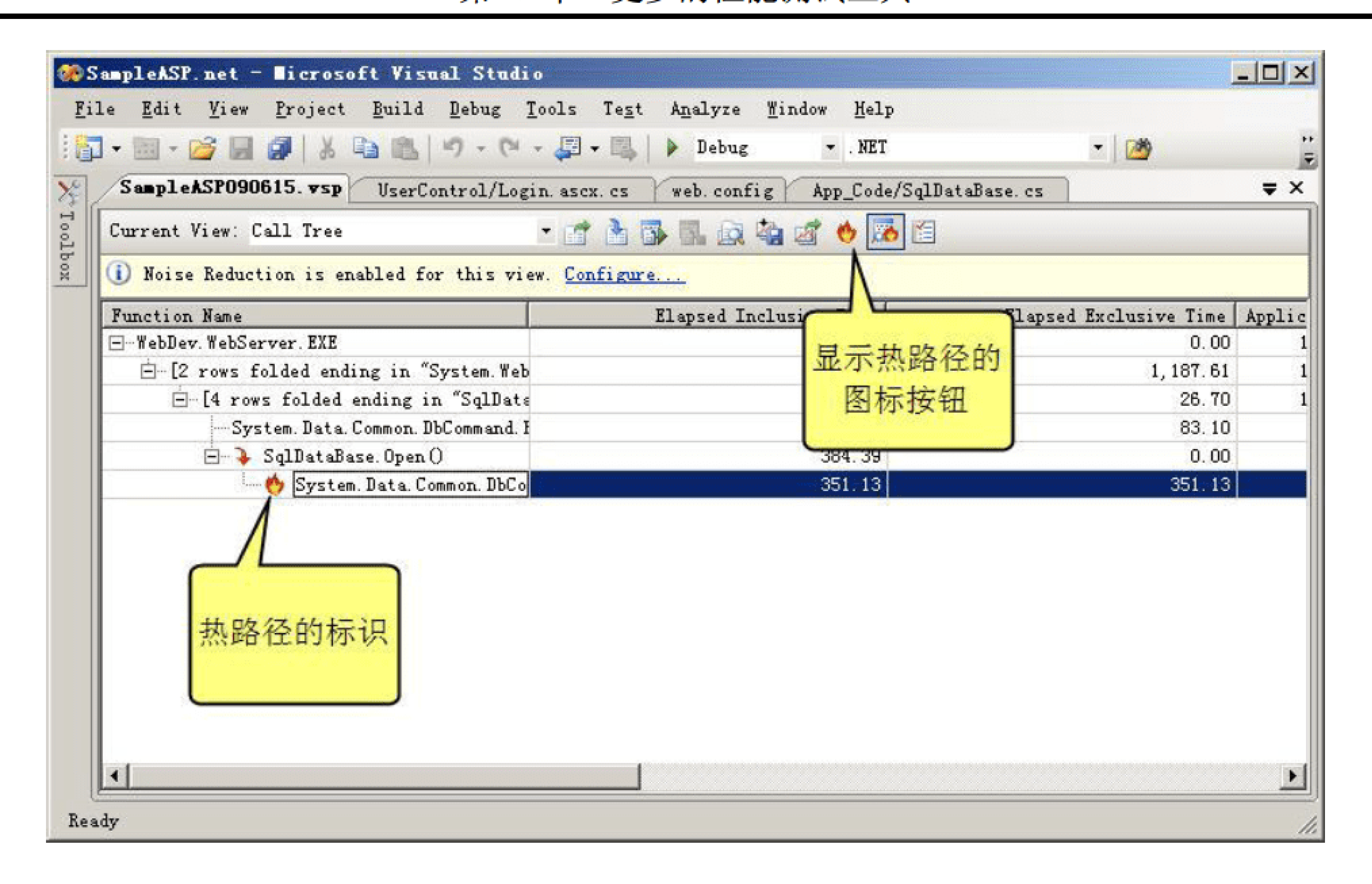


图 15-19　显示热路径

由于本节中笔者为了举例方便，所测试的 ASP.NET 代码非常简单，只包含了一个打开数据库连接的操作，因此热路径很短。实际情况中的热路径往往不会这样直白，需要细心和耐心去发现与确认。

通过调用树、内含时间与专有时间、热路径能使得测试工程师快速地在复杂的源代码中发现性能问题所在，因此是很有用处的。

分析器还有其他一些功能，但主要测试过程和重点在本节已经讲述了，就不再花费更多的篇幅，感兴趣的读者可以在实际项目中加以运用。

15.4　本 章 小 结

本章首先简要介绍了目前市场上的 3 类主流性能测试工具，便于读者了解 LoadRunner 之外各性能测试工具的总体情况。这 3 类性能测试工具分别如下。

- 企业级的性能测试工具：支持协议全面、功能强大、报表详细、价格也可能相对昂贵，如本书所介绍的 LoadRunner 等。
- 轻量级的性能测试工具：安装方便、易用、快速，如本章介绍的微软的免费测试工具 WAS。
- 白盒性能测试工具：能够发现代码中的性能问题，如本章介绍的微软的 Visual Studio 2008 分析器。

在本章随后的两节中，以微软提供的免费性能测试工具发展为脉络，介绍了 WAS 和分析器，并重点强调了采样法和检测法的区别，这两个名词绝不仅仅只存在于分析器的使用说明中：它们的区别有些类似黑盒测试与白盒测试的区别，也代表了性能测试工具的两大方向。用通俗的语言来描述，即是：

（1）从运行结果出发、面向用户体验的性能测试。回想 LoadRunner 的测试过程，可

以发现，我们对 Web 应用的代码知之甚少，只是从访问页面开始入手，获得相应的性能计数器指标，并对其进行分析。简单地说，就是从外部的性能表现发现内部问题的过程。大多数的性能测试工具都是以此为出发点的，并且能够更全面地发现 Web 应用中的性能问题。

（2）从执行代码出发、面向 Web 应用开发者的测试。在本章举出了 Visual Studio 2008 中分析器的例子。它可以使用检测法，对代码进行修改，记录函数执行时间等信息。简单地说，就是由内部的软件代码推及外部性能表现的过程。

第 16 章笔者将以某实际网站的性能测试为例，进行一次全过程的性能测试以总结前面章节所讲述的内容。

第 5 篇　Web 性能测试实战

第 16 章　大容量 Web 应用性能测试实例

本章以小白对公司开发网站进行性能测试的全过程为例，对前面章节介绍的测试知识与 LoadRunner 使用进行了总结。

Web 应用，特别是网站，对性能测试指标的考察主要是各项事务的响应时间。因此，小白所做的工作将主要围绕这类数据进行。

首先，需要对网站的应用背景有所了解，根据第 8 章的知识，它将决定性能测试具体针对哪些事务，在 LoadRunner 中建立哪些场景。

16.1　Web 应用背景

小白公司开发的网站采用 ASP.NET 技术，SQL Server 2005 作为数据库平台，IIS 作为网站服务器，是一个在实际工作中较常见的组合。目前，前端 IIS 应用了负载均衡技术，后端 SQL 数据库实现了故障转移群集。网络结构简图与网站逻辑简图与第 8 章的图 8-4、图 8-5 类似，这里就不单独列出了。

网站包含 4 大部分内容，分别如下。

- ❑ 新闻与业内动态：主要介绍公司以及行业内的新闻，为动态页面，内容从数据库中读取，注册用户还可以对新闻进行评论，但需要后台管理的审核通过才能显示在前台页面。
- ❑ 商城：提供了网上购买公司消费类产品的页面，注册用户可在线购买，支持网上支付也可以货到付款。
- ❑ 产品使用与支持论坛：注册用户可以下载产品更新、咨询产品问题、交流产品使用心得与技巧，另外还有其他一些相关通用话题的讨论区等。
- ❑ 后台管理区：主要对上述 3 大部分的内容进行管理，包括网站内容的上传、注册用户发布内容的审核、产品更新的发布、公司活动等其他信息的管理。

对于用户的管理，网站设定了如下几个规则：

（1）普通用户如果不注册，则只能浏览网站的内容而不能发布和在线购买商品。

（2）注册用户根据购买商品价值或者发精华帖的数量而具备不同的级别。

（3）注册用户可以直接修改自己的昵称、地址、付费方式等信息，但是对于新闻的评论需要管理员的审核后才能显示在页面上；在论坛的发帖和回复管理员有权删除。

在页面显示方面，为了提高客户端显示速度，网站技术部已经采用了一些方法，诸如：

（1）应用较多的层叠样式表（CSS）技术，减少每网页的下载字节数。

（2）采用共用的包含文件（Include File），特别是网站各页面包括导航条的上半部分。

（3）美工对图像进行了不影响显示效果的处理，合理选择图片的格式（一般来说，数

Gif 要比 Jpeg 文件尺寸要小，但只能包含 256 种颜色）。

（4）网页制作采取了页面内多个表格或层的编写方式，使得网页内容不是一次性显示而是渐次显示，间接缩短了响应时间，提高了用户体验。

在服务器端方面，技术部经理与网络管理员选择了合适的硬件以及条件较好的互联网数据中心（IDC）承载该网站，并采取了适当的 RAID、负载均衡、群集等技术。

据网站运营部、市场策划部等相关同事的统计，用户的行为集中分布在新注册用户、内容浏览、商城订单的生成、论坛的搜索这几个方面。

有了这些基本的背景知识，小白就开始起草有关性能测试计划文档，详见 16.2 节。

16.2　性能测试设计

进行 1 项新的性能测试之前，要计划好如下几个问题的答案。

由谁来做？做什么？在什么时间做？

以上问题都涉及性能测试中人员分工与时间计划这些资源的分配。小白必须在测试计划初稿中包含这样的内容，最终的分配则由技术部的经理来决定。

16.2.1　人员与计划

本次参与性能测试的主要人员与他们的职责如下：

（1）网站策划人员。网站策划人员主要负责网站各栏目、各子系统的流程介绍、与其他相关人员制订性能测试目标、根据用户数据确定关键用户操作。

（2）网站技术部经理。网站技术部经理负责确定性能测试进度、测试人员分工、测试过程中问题的解决、测试报告审核与性能优化讨论与决策。

（3）网站技术部开发人员。网站开发人员参与测试环境、测试数据、测试虚拟用户脚本的编写、提供 Web 应用的开发文档和必要支持。

（4）网站技术部测试人员。负责测试环境的搭建、测试方案与详细计划的编写、测试脚本的编写与调试、测试场景的配置、测试数据的准备、测试的执行、测试结果的汇总与分析、测试报告的编写。

（5）网站技术部网络管理人员。在运行测试的过程之前和当中，需要网络管理人员设置较好的网络和操作系统状态。在测试过程结束之后，需要网络管理人员一同发现服务器、网络的性能问题并进一步提出改进意见。

在确定了人员之后，就可以开始编写测试计划的部分。在前面的章节我们介绍了测试计划的主要部分，在这里根据实际情况进行编写，主要是测试的范围与测试目标的确定。

本次性能测试的测试对象是整个网站，目标是通过性能测试，发现可能的性能问题、为 Web 应用上线做出可靠性的评价并提出进一步优化的方案以便实施。

测试目标依据网站的各个栏目有所不同，具体列表如表 16-1 所示。表格中的优先级为 1 表示最高优先级，2 为次优先级，依次类推。

表 16-1　网站的性能测试具体目标

优先级	事务名称	预计每分钟发生平均数量（最小值 – 峰值）	性能目标
1	登录、注册新用户	0 - 100	CPU 平均使用率<=75%响应时间<3s
2	注册用户更新注册信息	0 - 10	CPU 平均使用率<=75%响应时间<5s
1	用户浏览新闻、行业动态内容	0 - 1000	CPU 平均使用率<=75%响应时间<3s
1	注册用户论坛发布、修改内容	0 - 10	CPU 平均使用率<=75%响应时间<5s
1	用户通过搜索功能查找产品更新	0 - 100	CPU 平均使用率<=75%响应时间<5s
1	注册用户浏览商城页面	0 - 100	CPU 平均使用率<=75%响应时间<3s
1	注册用户生成商城订单	0 - 10	CPU 平均使用率<=75%响应时间<3s
1	注册用户获得网上支付结果	0 - 10	CPU 平均使用率<=75%响应时间<3s
3	后台管理页面中保存内容	0 – 100	CPU 平均使用率<=75%响应时间<8s

【优先级】

这里所说的优先级，是指该事务或者场景在性能测试中的优先级，而不是其对于整个网站业务的优先级。举例来说，后台管理页面中保存内容这部分功能是非常重要的，它对于整个网站来说是需要优先实现的（没有它就不能快速、规范化地对网站增加/修改/删除内容，因此可能比某些前台页面还要早实现）。但对于性能测试来说，由于使用者为网站内部编辑人员，所处环境与生产环境也较近，因此列为性能测试中的较低优先级。

对于表 16-1 中的每分钟事务发生数量，则可以根据如下 3 个方面进行估计：

- 现有数据的最小值与峰值。这些数据可以是正在运行的旧版本网站的相应数据，或者是同类网站的相应数据。对于最小值，一般都是 0；对于峰值，可以根据现有网站浏览量的统计来得到。
- 未来数据。可以是根据网站目标制订的将来预计数据（比如，3 年内注册用户翻番等），也可以是同类网站中前茅者的现有数据。
- 理论数据。根据现有硬件条件与每个用户每次请求所下载的网页字节数相除得到。比如，网卡为千兆独享带宽，每个事务每次用户请求大约下载 1 兆数据，花费 10 秒钟，则每秒钟最多可以有 10000 次请求，1000 个事务。当然这是一个不准确的估计值，用于提供数值的大致上限。

【网站浏览量峰值的规律】

一般来说，网站浏览量峰值会出现在：周一的上午、工作日的中午、晚上至凌晨的几个小时等处。另外，刚刚结束某次网站宣传和推广活动，也可能会产生一次或多次峰值。凌晨 2 到 3 点之后以及周日，对于商业信息类网站的浏览量一般较少。周五以及周末对于网上购物网站来说，浏览量则会上升。总之，不同内容的网站其浏览量变化规律并不相同，需要在实际工作中发现。

【用户来源】

用户来源是另外一个重要的问题。比如来自教育网的用户访问公网服务器时所经历的时间会较长一些，很难用公网的响应时间来衡量。来自别的国家与地区的用户浏览网站时也可能会出现另外的问题（安装语言包、Flash 插件等都会大大增加页面的显示时间，从而影响用户体验）。这些问题往往较多较杂，一般以测试工程师手工运行一次或多次性能测

试为好，可以直接地找到原因。

有了人员与测试目标，就可以制订性能测试的具体时间表了，一般来说，性能测试的执行不宜花费过长的时间，这是由于：正式的性能测试往往处于项目进展后期，时间趋紧；长时间的性能测试会导致要分析的数据过多；再长的性能测试时间也无法代替网站运营时的真实数据，因此不必将测试时间拖得很长。

小白根据工作实际情况，列出了此次性能测试的详细时间表，如表 16-2 所示。

表 16-2　XX网站性能测试时间表

阶 段 名 称	起 止 时 间	阶段交付成果	参 与 人 员
测试准备（计划、目标和测试标准确定）	2008.10.8–2008.10.10	批准后的测试计划文档	测试经理、小白、需求分析部、开发工程师等
测试环境设计	2008.10.13-2008.10.14	测试环境文档	配置工程师
测试用例、工具脚本开发与调试	2008.10.15-2008.10.27	测试用例 LoadRunner 脚本	小白
- 用户注册、登录部分	2008.10.15 – 2008.10.16	LoadRunner 脚本	小白
- 用户浏览内容部分	2008.10.17 – 2008.10.17	LoadRunner 脚本	小白
- 用户商城订单部分	2008.10.20 – 2008.10.21	LoadRunner 脚本	小白
- 用户商城支付部分	2008.10.22 – 2008.10.23	LoadRunner 脚本	小白
- 用户社区发帖部分	2008.10.24 – 2008.10.24	LoadRunner 脚本	小白
- 后台管理更新内容部分	2008.10.27 – 2008.10.27	LoadRunner 脚本	小白
测试工具部署、测试场景创建	2008.10.28 – 2008.10.30	测试环境完成 测试部署文档 LoadRunner 场景文件	配置工程师、小白
执行性能测试	2008.10.30 – 2008.11.5	LoadRunner 测试结果文件	小白
执行压力测试	2008.11.6 – 2008.11.7	LoadRunner 场景文件 LoadRunner 测试结果文件	小白
测试结果分析、生成测试报告	2008.11.10 – 2008.11.12	测试分析报告	测试经理、小白
性能测试总结、优化讨论	2008.11.13 – 2008.11.13	性能测试总结、性能优化讨论会议记录	测试经理、小白、开发工程师等

16.2.2　测试环境的准备

测试环境要求与真实环境尽量相同。由于技术部内部调试网站代码还需要一套完整的系统，为了不影响其他同事的开发、测试工作，对于新上线的网站，可以在即将投入使用、现在尚空闲的生产环境中进行测试。当然，如果本次性能测试是针对网站的升级，那么现有生产环境是不能用来进行测试的，只能通过搭建相同配置的硬件或者虚拟环境来实现。

小白的网站采取了利用现有多余服务器搭建环境，模拟生产环境进行性能测试的方

法。这些服务器被用作技术部门的文件备份服务器，日常请求较少，配置如表 16-3 所示。

表 16-3　测试环境各服务器配置

服务器硬件配置	服务器用途
1U 的某品牌服务器 Intel Xeon 双核 3.0G 2G 内存 一块 73G 硬盘	Windows 2003 SP2 IIS 网站发布服务器 包含被测试网站的最新代码版本
1U 的某品牌服务器 Intel Xeon 双核 3.0G 2G 内存 两块 73G 硬盘	Windows 2003 SP2 SQL Server 2005 SP2 包含被测试网站现有的测试数据

【U 的概念】

针对网站的性能测试工程师有必要了解一下服务器高度单位 U。1U=1.75 英寸。为了节省空间，服务器一般层叠存放于机架内。因此，机架内部每隔一定高度会有固定的螺孔，类似日常生活中的板式家具。这样的最小高度间隔就叫做一个单位（Unit），简称 1U。有了标准化的 U 作为单位，各个品牌的服务器就可以并存于同一个机架内了。不同品牌、型号的服务器高度不同，可能是 1U、2U、4U 等甚至更高，而且一般来说，U 数多（即高度更高）的服务器性能要强，价格也更贵。

生产环境中所采用的负载均衡、故障转移等技术，可以在安装配套软硬件之后就进行测试，不在本次性能测试计划之内。另外，它们的主要目的是为了可靠性，总体而言对提高单次访问的响应时间并无很大帮助。因此，测试环境暂时不需要考虑这些配置，以节省环境配置时间。

生产环境中服务器的配置如表 16-4 所示。

表 16-4　生产环境各服务器配置

服务器硬件配置	服务器用途
2U 的某品牌服务器 Intel Xeon 四核 3.0G 32G 内存	网站生产环境数据库服务器
同上	IIS Web 服务器

【生产环境与测试环境之间的差异对测试的影响】

比较表 16-3 与表 16-4 二者所提供的环境，可见硬件差距主要在 CPU 与内存之中。在一个有限的用户数量之内，对单个用户（虚拟用户）所进行的网站操作而言，内存容量大小的影响可以忽略，而应该主要考虑 CPU 速度的快慢。但是，内存容量大小对于压力测试发现系统极限和并发用户数来说则是非常重要的。所以，虽然测试环境与生产环境硬件配置不同，响应时间的具体数值还是具备很大的参考意义，其变化规律也基本一致（相同的网站代码下比较），而最大连接数等并发数据则会有显著的不同。

由于小白采用了 LoadRunner 进行性能测试，因此需要有压力生成器来产生工作负荷。这些压力生成器相当于真实生产环境中的用户客户端电脑，如表 16-5 所示。

表 16-5　性能测试中的压力生成器

序号	硬 件 配 置	安装操作系统与软件
1	1 个 CPU，1G 内存	Windows XP SP3，IE7，LoadRunner 压力生成器
2	1 个 CPU，1G 内存	Windows XP SP3，IE8，LoadRunner 压力生成器
3	1 个 CPU，2G 内存	Windows Vista SP1，IE7，LoadRunner 压力生成器
4	1 个 CPU，4G 内存	Windows Vista SP1 64bit，IE7，LoadRunner 压力生成器
5	1 个 CPU，512M 内存	Windows XP SP2，IE6，LoadRunner 压力生成器

在表 16-5 中尽量安排了几种不同硬件软件的组合，以模拟真实情况。有了压力生成器，还需要有控制器来连接它们。这个任务就坐落在小白的一台工作机器上了，它的配置为 1 个 CPU，2G 内存，Windows XP 系统。

【控制器与压力生成器】

作为控制器的电脑与作为压力生成器的电脑必须能够相互访问，最好采用相同的用户名登录。在控制器中增加压力生成器的时候可以进行连接测试。

至此，在测试环境方面就搭建完毕了，下面将进行测试场景的详细规划与脚本录制的步骤。

16.2.3　测试场景的设计

在表 16-2 中小白列举了性能测试中必须完成的事务列表，本节将以其中的用户登录和网上支付两个典型事务为例进行 LoadRunner 场景的设计。

【场景设计先于脚本录制的原因】

在前面介绍 LoadRunner 使用方法的章节中，脚本是先于场景进行讲解的，这是因为在软件中，没有脚本，场景就无法设置和执行。在实际工作中，不必拘泥于这样的顺序，而应该从实际出发，综合考虑被测试的事务，在把一切问题难点都解决之后再来录制脚本和调整脚本，往往能够得到好的效果。另外，提前考虑场景有利于发现问题，而这些问题通常是网站其他人员（比如网站内容与策划人员对于商业流程的理解，开发人员对于代码内部特点的理解）理解的更为透彻，早些集中请教他们会更好。

如表 16-6 列出了用户登录场景的设计要求。

表 16-6　用户登录场景要求

项　　目	项目具体说明
该场景下的操作	用户通过该场景，输入用户名、密码以及验证码登录网站。如果三项匹配成功，则显示成功并跳转到相关后续页面；否则，返回登录页面
场景测试步骤与目标结果	通过单个虚拟用户的迭代多次登录行为，获得网站在单用户访问时的平均响应时间与各图表监控数据。 通过多个虚拟用户的迭代多次登录行为，获得网站在多用户访问时的平均响应时间与各图表监控数据。 通过改变并发用户数，获得网站在不同并发数量用户访问时的平均响应时间与各图表监控数据

续表

项　　目	项目具体说明
场景测试分析方法	通过单个用户与多用户访问此场景的平均响应时间的比较，考察网站在用户增加后响应时间的变化情况。 通过多用户的图表监控数据变化，考察整个系统在此场景下的性能变化规律。 考察不同并发数对于性能的影响
场景配置	根据实际情况，用户登录的思考时间设置为较小值。 迭代之间的间隔设置为 0。 对于多用户的测试，采用渐进式的虚拟用户增加方式。 对于并发用户的测试，根据表 16-1 的数据，并发用户数分别设置为确良 50、100 和 200

如表 16-7 列出了网上支付场景的设计要求。

表 16-7　网上支付场景要求

项　　目	项目具体说明
该场景下的操作	用户选择商品后，选择网上支付选择，输入用户名、密码以及验证码登录网站。如果三项匹配成功，则显示成功并跳转到相关后续页面；否则，返回登录页面
场景测试步骤与目标结果	通过单个虚拟用户的迭代多次登录行为，获得网站在单用户访问时的平均响应时间与各图表监控数据。 通过多个虚拟用户的迭代多次登录行为，获得网站在多用户访问时的平均响应时间与各图表监控数据。 通过改变并发用户数，获得网站在不同并发数量用户访问时的平均响应时间与各图表监控数据
场景测试分析方法	通过单个用户与多用户访问此场景的平均响应时间比较，考察网站在用户增加后响应时间变化情况。 通过多用户的图表监控数据变化，考察整个系统在此场景下的性能变化规律。 考察不同并发数对于性能的影响
场景配置	根据实际情况，用户登录的思考时间设置为较小值 迭代之间的间隔设置为 0 对于多用户的测试，采用渐进式的虚拟用户增加方式 对于并发用户的测试，根据表 16-1 的数据，并发用户数分别设置为 50、100 和 200

根据这样的要求，再进行脚本的录制与修改目的性就比较明确了。

16.2.4　测试脚本的录制

本节对前面所述的用户注册和网上支付两个典型事务，录制 LoadRunner 脚本中可能遇到的难点进行介绍。

1．录制用户登录脚本时需要考虑的问题

用户注册场景主要的难点在于：

- ❑ 验证码的变化。对于每一个虚拟用户登录系统的过程，网站给出的验证码一般都是不一样的，为了脚本能够顺利地进行而不出现全体登录失败的错误，设计场景时要考虑解决验证码的干扰问题。
- ❑ 用户名有效性的验证。对于一个测试中的虚拟用户，如何确保它使用的网站用户名不被别的虚拟用户使用呢？

2. 验证码问题的解决

关于验证码，小白的解决方案有如下 3 种：

1）简化法，即取消验证码的方法

网站开发人员可以临时将代码中生成和比较验证码的部分注释，然后编译成一个测试的网站版本，部署在测试环境中。这样的解决办法虽然方便了性能测试的执行，但还是增加了测试环境与生产环境中的差异，忽略了验证码可能带来的性能问题（如果多用户同时注册或登录，验证码可能显示不出来，导致登录失败）。另外，在性能测试通过、网站上线之后，如果要再次进行用户登录部分的性能测试，还要修改脚本，维护成本上升（生产环境中取消验证码是不安全的，无法实施）。因此，取消验证码的方法是一个只能用于测试环境且在时间较紧的情况下才使用的方法，并不推荐。

2）信任法

网站开发人员不取消代码中生成和比较验证码的部分，而是指定一个万能的验证码，并且加入 IP 地址的判断来确保安全，这样就可以既保证测试环境和生产环境中的代码一致，还能保证安全和性能测试的方便。该方法的具体步骤如下：

（1）技术部经理、开发人员和测试人员确定一个万能验证码。

（2）开发人员在网站源程序中验证码相关代码部分，加入对万能验证码的判断。判断条件是：如果输入了万能验证码，再加之其请求的 IP 地址处于公司局域网内的范围，就可以放行。

（3）测试工程师修改录制后的脚本，将其中验证码都修改为万能验证码的值。

之所以采用 IP 地址判断，是因为如果万能验证码流传开来，对安全将是很大的隐患；而公司内部局域网的 IP 地址所处网段基本一致，将可以输入万能验证码的用户限制在此范围内较容易控制。当然，也可以采取别的方法对能够使用万能验证码的用户加以限制。

3）一劳永逸法

该方法通过编程实现当前验证码的获取。开发人员将这样的功能封装为一个动态链接库 DLL，将其复制到 LoadRunner 的 bin 目录下。测试工程师在脚本中利用 LoadRunner 函数 lr_load_dll()（DLL 文件名）调用它即可，如代码 16-1 所示。

代码 16-1　在脚本中调用外部 DLL 代码片段

```
...
Ret = lr_load_dll("AuthenticateCode.dll"); // 加载外部的验证码 DLL 文件

If(Ret !=0)
{ // 加载失败的情况，将错误信息写入 LoadRunner 日志
    Lr_output_message("[Error] Load AuthenticateCode.dll error! Please
have a check!")
}
```

```
Else
{
    GetCurrentAuthentiCode(code);

    web submit data("login.aspx",
        "Action=http://192.168.1.253:1080/Users/login.aspx",
        "Method=POST",
        "RecContentType=text/html",
        "Referer= http://192.168.1.253:1080/",
        "Snapshot=t2.inf",
        "Mode=HTML",
        ITEMDATA,
        "Name=userSession", "Value={WCSParam Diff1}", ENDITEM,
        "Name=username", "Value=xiaobai", ENDITEM, // 用户名
        "Name=password", "Value=xiaobai", ENDITEM, // 密码
        "Code=authenticode", code, ENDITEM, // 将前面获得的验证码作为输入
        "Name=login.x", "Value=49", ENDITEM,
        "Name=login.y", "Value=2", ENDITEM,
        "Name=login", "Value=Login", ENDITEM,
        "Name=JSFormSubmit", "Value=off", ENDITEM,
        LAST);// 登录系统
...
```

如表 16-8 列出了这 3 种解决验证码问题的对比作为总结。

表 16-8　3 种在性能测试中解决验证码问题的比较

方　　法	优　　点	缺　　点
简化法 （在性能测试中去除验证码功能的方法）	方便，测试工程师不必考虑验证码，开发人员修改代码代价小	不安全。 只能应用于测试环境。 增加版本维护成本
信任法 （在控制范围内使用万能验证码）	方便，测试工程师在脚本中指定验证码，开发人员修改代码代价小	较安全
一劳永逸法（调用外部 DLL 文件获取当前验证码）	安全	测试工程师和开发人员的代码代价较高。 增加版本维护成本

这 3 种解决验证码的方法可以根据实际情况选择使用，读者还可以创造出新的方法。

3．用户名有效性问题的解决

关于用户名有效性的问题，小白采用了这样的办法：

（1）预先在测试数据库中生成一系列的用户名，比如 Vuser1、Vuser2……等直到 Vuser1000，这可以通过编写一段包含循环的简单脚本（比如 asp，javascript）来实现，如代码 16-2 所示。

代码 16-2　批量增加网站测试用户的 asp 代码片断

```
...
'数据库链接代码
set objConn=server.CreateObject("ADODB.connection")
connStr="PROVIDER=SQLOLEDB;Server=(local);DATABASE=testData;UID=sa;PWD=
sa;"
objConn.Open connstr
```

```
'取得用户名前缀、形如 Vuser1、Vuser2……中的 Vuser.可自行在页面设置
valName = Cstr(Request.Form("UserNamePrefix"))

'执行插入数据库的过程, 1001 次
For i = 0 to 1000
 '插入数据库用户表的 SQL 语句
 valUserName = valName + i
 objConn.execute("insert into Users(username,pwd) values('"+valUserName+
 "','" +valUserName+ "')"
Next

Response.write("数据插入成功")

'关闭数据库连接
objConn.close
Set objConn = nothing
...
```

（2）利用 LoadRunner 自带的函数 lr_whoami()来获得当前运行脚本的虚拟用户 id，并将该数字与字符串 Vuser 拼接，以生成的用户名（形如 Vuser1）登录网站。我们知道，在第 1 步中，这些用户名已经在数据库中存在了。Lr_whoami()的使用方法如代码 16-3 所示。

代码 16-3　Lr_Whoami()函数使用说明

```
...
// id 和 scid 都是 int 类型
int id, scid;
char *vuser_group;
// 调用函数，获得三个输出参数的值
lr_whoami(&id, &vuser_group, &scid);
// 在 LoadRunner 中显示出来
lr_message( "Group: %s, vuser id: %d, scenario id %d", vuser_group, id, scid);
```

当然，也可以所有的虚拟用户都是用同一个用户名进行登录，这样就不必使用前面的 2 个步骤了。但是，对于有些网站，为了防止恶意下载网站内容等原因，它限制了一个用户登录的次数和时间间隔，在这种情况下，为每个虚拟用户名创建一个网站用户就很有用了。

4．网上支付问题的解决

注册用户选择好商品、生成订单之后，要进入支付页面进行方式选择。目前，大多数网站的支付方式为货到付款、网上支付等。除网上支付之外，其他方式被选择后注册用户关于商品购买的操作就结束了，因此这些方式的性能测试与其他网页的性能测试并不相同。网上支付则有些特别，它需要调用银行提供的支付网关，小白在测试中主要考虑了如下两个问题。

- 如何仅仅测试网上支付，而不是真的把订单金额付款给银行？
- 如何获得支付是否成功的信息？

对于第一个问题，其实很好解决。银行在提供网上银行的支付网关时，都会提供一个测试接口，可以直接调用这个接口即可，不必担心测试人员工资卡内的金额会随性能测试

的进行而逐渐减少。在测试的过程中，如果出现问题，还可以直接和银行负责这方面的部门人员进行联系解答。

对于第 2 个问题，可以使用 LoadRunner 函数 web_reg_find()在网页中进行字符串的搜索。网上支付完毕后，浏览器将从银行支付网关转向网站提供的结果界面，成功或者错误都将在该结果页面上反映出来。因此，使用字符串搜索函数 Web reg find()，在支付结果页面中查找类似“支付成功”或者“支付失败”的文字，即可获得网上支付成功与否的结果。

Web_reg_find()函数的使用示例如代码 16-4 所示。

代码 16-4　Web_reg_find()函数使用说明

```
...
// 访问支付结果页面
web_url("MercuryWebTours",
     "URL=http://192.168.0.253/shop/epayResult.aspx",
     "Resource=0",
     "RecContentType=text/html",
     "Referer=",
     "Snapshot=t1.inf",
     "Mode=HTML",
      LAST);

// 设置页面中查找的字符串
web_reg_find("Text=支付成功", LAST);
...
```

当页面中未找到指定字符串时，脚本将会失败。这种方法在验证注册用户是否登录成功等方面也很有用处。

5. 总结

以上本书以用户登录和网上支付两个场景为例，介绍了脚本录制与改进中可能涉及的具体问题。在实际工作中，还可能有更多的问题出现，从本节的介绍可以看出，依靠 LoadRunner 自带函数是解决这些问题的不错选择。

【单元场景和综合场景】

运行单个场景对于 Web 应用的性能测试是远远不够的，因为在同一时刻，有的用户在进行登录，有的用户在进行网上支付，有的用户在浏览页面等，各个场景对于系统的整个性能表现都同时起着作用。为了更好、更真实地模拟实际情况，有必要把这些场景集成在一起，形成一个或多个综合的场景，在其中不同的虚拟用户组执行不同的场景。这样的层次关系很类似功能测试中的组件测试与集成测试，可以称之为单元场景（完成单一功能）和综合场景（同时完成多项功能）。

16.2.5　测试监控设置

小白此次性能测试中的 LoadRunner 监控设置并未超出前文介绍过的各项 Web 应用图表监控范围，但是与前面章节不同的是，针对不同场景，小白根据其特点，使用了不同的监控度量。表 16-9 列出了其中一个典型场景的监控设置，该综合场景包含了多个 16.2.4 节中的单元场景。

表 16-9　某典型场景的LoadRunner监控设置

场 景 名 称	场 景 安 排	监 控 设 置
最热新闻 （在某个重要日期，众多用户访问同一新闻网页的情况）	各单元场景及其虚拟用户数量： ❑ 用户登录　100 ❑ 新闻浏览　200 ❑ 生成订单　50 ❑ 网上支付　20 用户登录、生成订单、网上支付都采取场景设置中的渐进式（Ramp Up）用户增长模式，每 10 秒增加 2 个。 除此之外，新闻浏览单元场景的脚本中增加一个集合点，设置为 50，以模拟众多用户查看同一条最新新闻的情况。 场景的迭代时间间隔为 0 秒、 运行时间为 2 个小时	❑ 每秒钟事务数量 ❑ 网络吞吐量 ❑ 每秒事务错误 ❑ 当前运行的虚拟用户数量 ❑ 数据库相关的性能计数器 ❑ CPU 处理器时间 ❑ 可用内存数量 ❑ IIS 相关的性能计数器

通过对各种单元场景进行有针对性、包含不同侧重点的组合，就能够模拟出不少真实运行的情况，有利于更全面地获得接近真实的性能数据，利于性能测试结果分析。

在设置好单元场景和综合场景之后，就可以开始执行性能测试了。

16.3　执行性能测试

一般而言，性能测试的执行需要按照如下的顺序：

（1）通过单虚拟用户运行单元场景来调试脚本和获得基本数据。

（2）通过多虚拟用户运行单元场景来获得多用户下该场景的性能数据，与第 1 步的数据相比较，可以得到多用户对于性能的影响。

（3）通过多虚拟用户运行综合场景，来获得多用户下各单元场景混合后 Web 应用的性能数据，与第 2 步的数据相比较，可以得到多场景，对于整体性能的影响。这时得到的结果已经接近于真实情况了。

（4）通过不断增加虚拟用户执行综合场景的方法，发现 Web 应用稳定性的极限，即压力测试。

由于执行性能测试涉及的步骤和测试结果较多，可以采用一些文档进行记录，比如表 16-10 列出的性能测试记录。

表 16-10　性能测试记录

测 试 内 容	多虚拟用户执行单元场景		
测试人员/日期	小白 / 2008-10-18	测试场景	用户登录
测试目标	平均响应时间遵照测试目标文档所要求，小于 3 秒		
场景设置	200 个虚拟用户渐进式增加，每 15 秒增加 4 个，思考时间 3 秒钟、迭代时间为 30 秒，总共执行 2 个小时		

续表

测 试 内 容	多虚拟用户执行单元场景
测试结果	每秒事务数量：21.3 事务平均响应时间：2.94 事务成功率：86%
测试性能监控	IIS：CPU 使用率峰值 63% 平均 15% SQL Server 2005：CPU 使用率峰值 55% 平均 40% ……
测试基本结论	虚拟用户小于 150 时，对系统性能无很大影响。 当增加到 200 左右时，数据库资源占用上升较大。 由于使用了缓存等各种技术，IIS 在初次访问时资源占用较大，之后占用较小。 总体响应时间基本符合要求。 失败的事务多数为连接超时等原因
测试详细结果	\\TestServer\PerfResult\20081019-Full\Senarios\UserLogin

通过类似表 16-10 这样的记录，可以对众多的测试数据和结果进行管理，便于分析和日后总结。这些记录可以放置在一个单独的文件夹内，以供查看。

具体执行性能测试的过程由 LoadRunner 控制器来实现，这点在前面的章节中已经介绍过了，就不再赘述。

16.4　测试结果与分析

执行完性能测试之后，将打开 LoadRunner 的分析器（Analysis）对结果进行分析。按照第 14 章的内容，首先需要验证测试结果的有效性。

与通用测试结果（指用自行开发工具进行测试后的结果）的验证不同，验证 LoadRunner 测试结果有效性的方法主要是查看分析器概要报告（Summary Report）中的 HTTP 反馈概要表。该表列出了测试过程中各种 HTTP 返回码的数量，如果其中没有或者很少出现 4XX，5XX 这样的错误代码，则说明测试有效。进行这样的判断主要是为了避免如下情况的发生：被测试 Web 应用或者 LoadRunner 脚本发生错误，导致测试结果无效。无效的数据会干扰性能测试工程师对正常结果的分析。

Web 应用的性能问题可以从服务器、网络和软件 3 方面来考虑：

- ❑ 服务器的性能问题主要是指操作系统、数据库、应用服务器的配置不当。
- ❑ 网络的性能问题主要是指服务器与客户端之间设备的配置问题。
- ❑ 软件的性能问题主要是指 Web 应用的代码考虑不周、存在各种 Bug。

下面三节将对这 3 个方面进行讲解。

16.4.1　发现服务器问题

从 LoadRunner 监控图表结果发现服务器存在的性能问题，可使用如下步骤。

（1）将当前运行的虚拟用户图（Running Vuser Graph）与系统资源图（System Resource

Graphs）中的 CPU 使用率等计数器结果曲线进行合并（Merge Graph）显示，以发现整个系统是否符合基本要求，比如 CPU 使用率小于等于 75%。

在 LoadRunner 分析器的测试结果中，打开当前运行的虚拟用户图和 Windows 资源图。由于涉及合并图表（Merge Graph）中的关联（Correlate）操作，首先要将 Windows 资源图中除 CPU 使用率指标之外其他性能计数器的数值过滤掉。如图 16-1 所示。测试工程师通过在资源图空白处右击，在弹出的快捷菜单中选择 Filter（过滤）命令，将使得图表中只包含 CPU 使用率的数据。

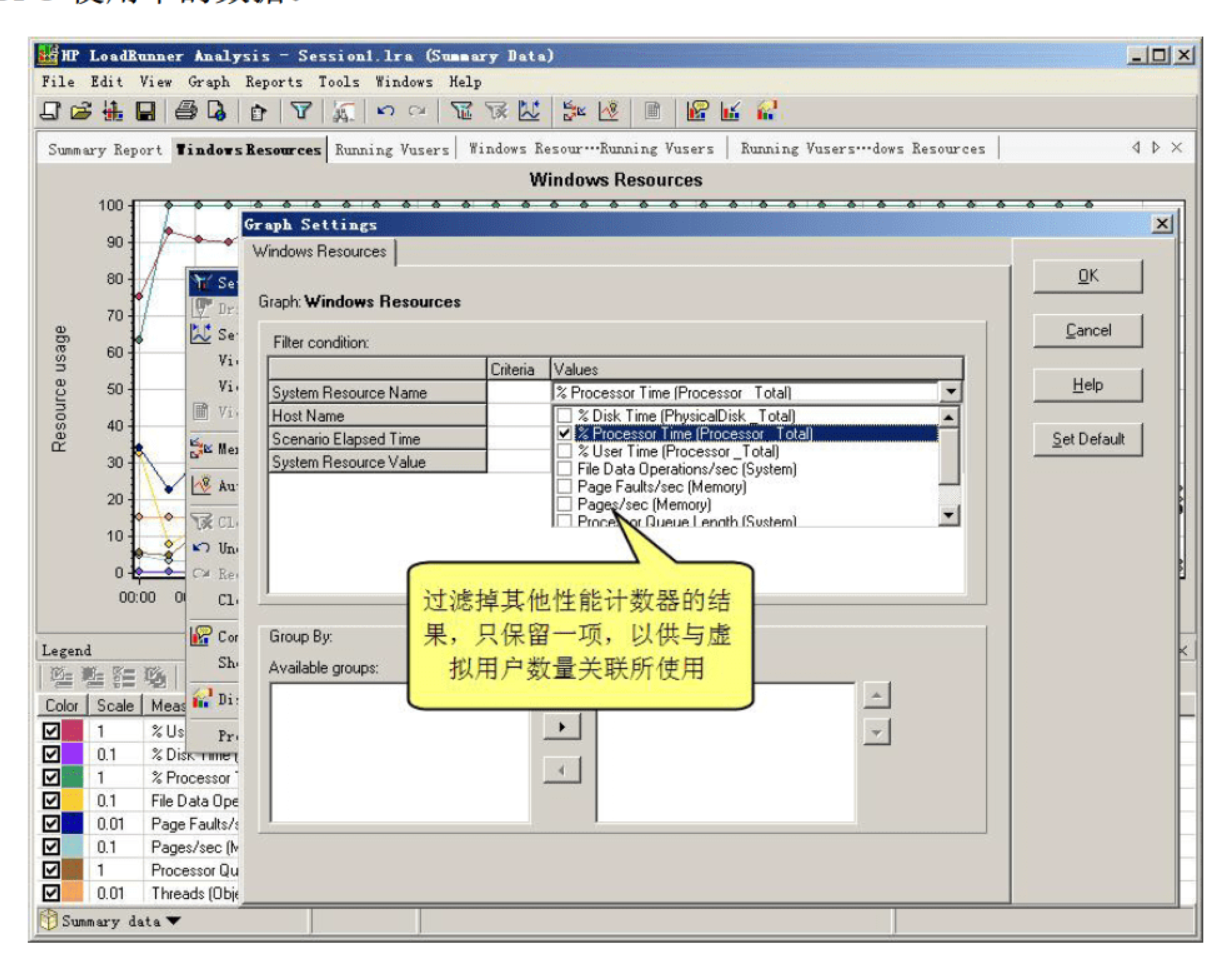


图 16-1　合并图表前对 Windows 资源图进行指标过滤

设置完毕后，在 Windows 资源图空白处右击，在弹出的快捷菜单中选择 Merge Graph（合并图）选项，在弹出的合并方式中选择 Correlate（关联）选项即可生成以虚拟用户数量为 X 轴、CPU 使用率为 Y 轴的图表，如图 16-2 所示。

从图 16-2 中发现，在虚拟用户大约为 7 的时候，整个系统的 CPU 使用率已经在 75% 之上了，因此可以说当前被测试的系统是存在很大性能问题的。当然，这个图表是本书所列举性能未达标的范例（以一台普通的笔记本作为服务器），在实际工作中，是不会用性能如此差的机器担当服务器的重任的。

总而言之，通过系统资源图中的各性能计数器曲线与虚拟用户数量进行关联，可以确定 Web 应用基本性能是否达标。

（2）采用类似的方法，将虚拟用户与事务图进行关联，可以发现事务的响应时间等是否符合要求。

以上所发现的性能问题都是将服务器作为一个整体来考虑的，获取的是操作系统性能计数器指标。本节之初，我们知道，服务器性能问题还包括数据库、应用服务器配置带来的问题，这就需要在执行性能测试的时候添加数据库与应用服务器的计数器了。在获得测

试结果之后，也可以将这些计数器分别与虚拟用户数量等进行关联，从中发现性能瓶颈所在的具体位置。

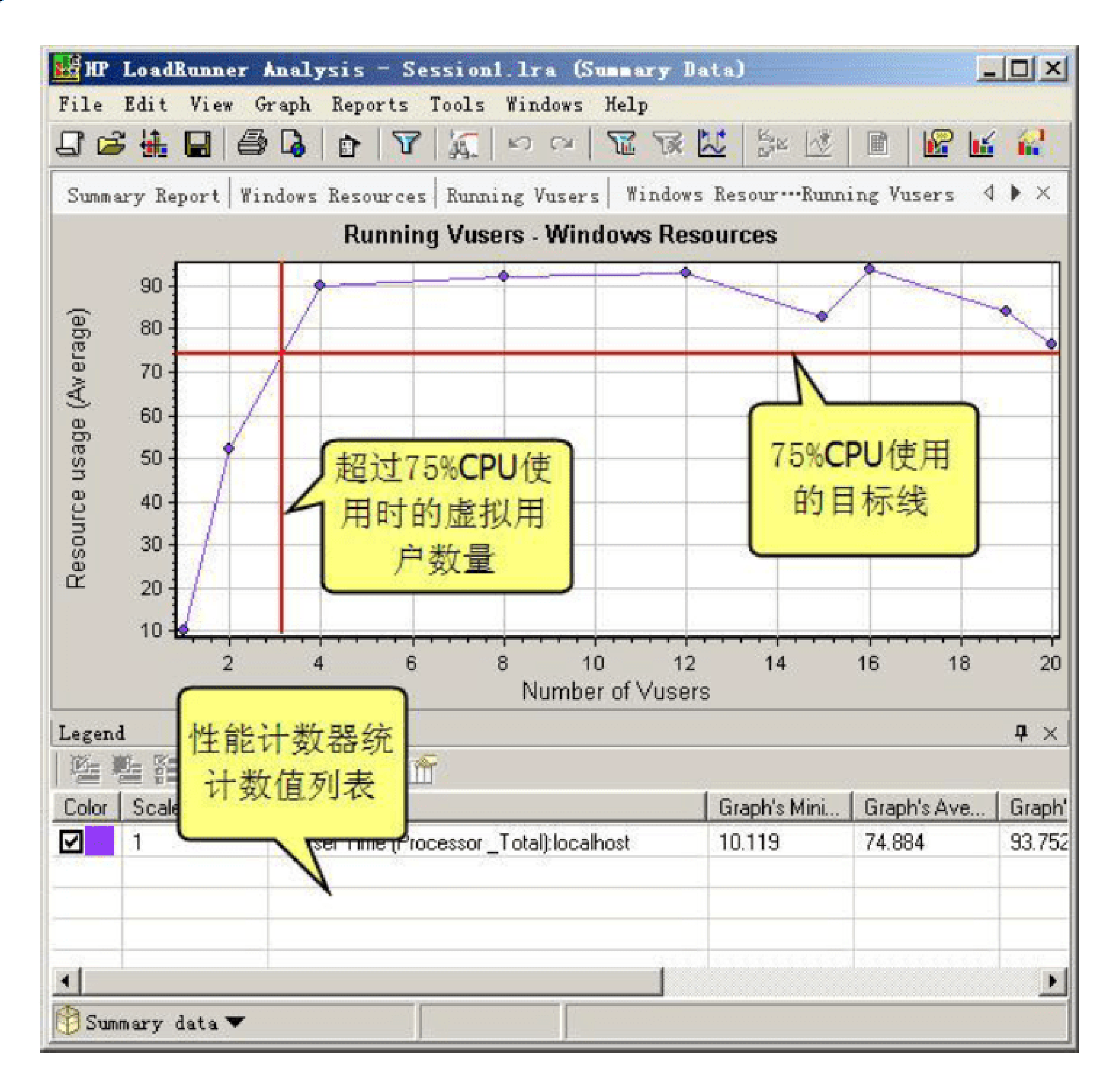


图 16-2　CPU 使用率与虚拟用户数量的关联图

【吞吐量图的作用】

网络资源图中的吞吐量图对发现应用服务器的瓶颈起到很大的作用。当吞吐量与用户增长规律相反时，就预示着问题的出现。这样的问题大部分是由于应用服务器的问题，也有小部分可能是 Web 应用代码的问题。

一般而言，不考虑 Web 应用代码的问题，数据库本身出现性能瓶颈的原因可能有下面几方面

- 索引设计不合理。
- 内存容量受限制。
- 数据库表字段设计不合理。
- 数据量大并且没有采取一定措施进行分库分表。
- 数据库配置问题。

如果需要快速、准确地发现数据库瓶颈的原因，读者可以阅读数据库优化相关的书籍。

除去 Web 应用带来的问题，应用服务器出现瓶颈的原因可能有：

- 应用服务器配置不当，比如缓存、超时时间、保持连接等各种属性的设置等。
- 系统资源比如内存与 CPU 性能不足。

16.4.2　发现网络问题

对于发现网络带来的性能问题，前文提到的分析器 HTTP 响应码列表可以算是一个途

径。不过，主要还是通过 Network Resources Graphs（网络资源图）和 Web Page Diagnostic Graphs（网页调试图）这两类图表进行分析。Web 应用中的文件从服务器端下载并显示在客户端可以划分为多个时间段，它们分别如下。

- DNS 解析时间（DNS Resolution Time）：用户发送 HTTP 请求后，DNS 服务器将请求的 DNS 名称解析为 IP 地址并返回所需的时间。
- 连接时间（Connections Time）：与指定 URL 的 Web 服务器建立初始连接所花费的时间，如果该项数值较大，则需要考虑网络连接问题。
- 第一次缓冲时间（First Buffer Time）：与 Web 服务器连接后，用户发出请求到成功接收来自服务器的第一次缓冲所花费的时间。
- SSL 握手时间（SSL HandShaking Time）：建立 SSL 连接所花费的时间。它包括客户端呼叫、服务器呼叫、客户端与服务器端公钥证书传输等子步骤。当采用 HTTPS 协议进行通信时该项数值才不会为 0。
- 接收时间（Receive Time）：从服务器完成接收数据过程所花费的时间，字节数与时间比值可用于评估网络传输质量。
- FTP 验证时间（FTP Authentication Time）：该时间只有在采用 FTP 协议进行通信时数值才不为 0。显示了 FTP 服务器验证客户端所花费的时间。
- 客户端事件（Client Time）：客户端用户所花费的时间，包括在页面的思考时间等各种情况。
- 错误时间（Error Time）：从发出 HTTP 请求到返回错误信息所经过的平均时间。

可以通过网页下载时间细分图来获得这些时间数值，对于测试过程中每一种时间的突然变大，都可以作为发现性能瓶颈的突破口。

16.4.3　发现软件代码问题

软件代码问题是指开发人员所编写的不完善代码给 Web 应用带来的性能问题，主要有如下几个方面：

- 服务器资源的不合理使用。比如内存的申请与释放等。
- 采用了不合理的数据结构或者陈旧的调用。
- 数据库查询语句编写上的不合理。
- 网页文件尺寸过大。
- 代码逻辑的缺陷。

在 LoadRunner 中，网页调试图可以列出每个网页所花费的时间，从中可以发现造成响应时间较长的网页；系统资源图可以列出页面文件、内存使用等服务器资源和数据库连接等性能计数器的使用情况；网络资源图可以发现文件尺寸对于响应时间的影响等。

对于通过性能测试发现软件代码问题，测试工程师需要和开发人员协同工作，确定问题的类型，之后修改和优化代码的工作将由开发人员来完成。对于更有经验的测试工程师来说，如果手头有代码，还可以使用类似第 15 章所介绍的白盒测试工具来发现问题代码的具体位置。

16.5　测试报告的生成

在获得了测试数据并分析出结果之后，就进入完成测试报告的阶段。之所以用“完成”这个词语，是因为测试报告的很多部分实际包含了测试计划、测试记录的内容，可以将这些文档的部分复制到报告中来。

在此特别强调一下测试报告的编写要领，它们分别是：

- 首先要能判断测试数据的准确性。根据本章介绍的几种经验规则对数据进行筛选。
- 预先估计性能测试数据的大致状况。有了大致性能好坏的估计，对于报告结论就会有较清晰的认识。性能的大致信息可以通过本章介绍的一些统计学小知识来获得。
- 对性能测试报告进行细致的分析。
- 讲究技巧、注重逻辑性和易读性，通过文字、数据与图表的有机结合，编写测试报告。
- 与测试经理和其他相关人员进行沟通，最终完成测试报告。

【图表数据的处理技巧：利用比例缩放】

对于图表数据，如果同一张图表内，各数据不在同一个数量级上，原封不动地显示会造成某类数据的变化规律不明显。我们可以通过将这类数据乘以或除以某个数值，使得它们与图表中其他数据处于同一数量级，这样显示出来的规律性更容易被读者把握。在 LoadRunner 中也有这样的方法，比如选中 View Graph Trends（查看图表趋势）菜单等操作。

性能测试报告应该包括如下几大部分：

- 性能测试的基本信息： Web 应用背景简述、测试人员、工具列表等必要信息。
- 性能测试的目的和目标：列出性能测试指标，可能的话，列出各场景的不同测试目标。
- 测试方法与计划：列出所使用的测试工具、基本流程、时间人员的安排等。
- 测试数据概括总结：测试数据的获取方法、数据有效性验证等。
- 测试结果分析：对于测试结果的重点分析。
- 测试结论：对本次测试结果进行总结。
- 测试优化建议：根据测试结论，可以提出一定的优化建议。

通过对数据进行认真、严谨的采集与分析，并坚持前文的若干原则，适当使用一些技巧，相信每个人都可以编写出一份出众的测试报告，获得工作上的承认和成就感。

16.6　本 章 小 结

本章通过小白对于公司网站进行一次完整性能测试的实例，将前面章节讲述过的各个知识点进行了串联，相信读者会加深利用 LoadRunner 进行性能测试的理解。

对于普通的测试而言，需要有一系列的测试用例输入到被测试软件当中，比如当用户输入非法字符时，系统能否提示错误这样的情况。而对于 LoadRunner 性能测试来说，这些

测试用例可以归为如下几类：

- 单个用户访问 Web 应用中的某个功能。
- 多个用户访问 Web 应用中的某个功能。
- 并发用户访问 Web 应用中的某个功能。
- 多个用户访问 Web 应用中的多个功能。

从以上分类出发，LoadRunner 的每个脚本就能够看成一个测试用例，而场景则是测试用例的组合。对于其他的性能测试工具，也具备类似的特点。可以看出，其实性能测试与其他类型测试一样，都在于考察不同的数据和行为组合下，被测试软件的表现。

本书进行到这里，有关性能测试的具体内容就基本结束了。但是，由于性能测试的目的在于性能优化，因此，在第 17 章，我们将以 ASP.NET 开发的 Web 应用为例，介绍一些性能优化的知识，使得整个过程更符合实际的需要。

第 17 章　Web 性能优化

前面的章节都是围绕 Web 性能测试来展开的，我们知道，性能测试的最终目的并不只是获得一些数据，而是从这些数据出发，为性能优化的决策提供最直接的参考。在本章中，我们将通过一些具体的技术来讲述若干 Web 性能优化的技巧。

一个 Web 应用系统，一般来说可以分为表示层（前台页面等）、中间层（负责业务逻辑等）以及数据库（负责存储 Web 应用数据）3 大部分。因此，对于 Web 应用的性能优化，也需要从以上 3 个方面来考虑，由于表示层与中间层均包含程序代码，为简单起见，本章把它们合二为一，讲述对代码与数据库的优化过程。

17.1　Web 应用代码的优化

小白公司开发的网站采用 ASP.NET 技术，SQL Server 2005 作为数据库平台，IIS 作为网站服务器，这也是一个在实际工作中较常见的组合。目前，前端 IIS 应用了负载均衡技术，后端 SQL 数据库实现了故障转移群集。网络结构简图和网站逻辑简图与第 8 章的图 8-4、图 8-5 类似，这里就不单独列出了。

17.1.1　ASP.NET 页面的优化原则

ASP.NET 开发的页面在如今的网站中占有很大比例，小白公司所开发的就是其中一例。简单地说，优化 ASP.NET 页面有如下的若干窍门。

- 节约原则：若页面中不使用某对象，则可以将该对象关闭。
- 恰当原则：合理利用各种数据结构、平台所提供的方法来提高性能。
- 缓冲原则：如果可能，尽量使用缓存、服务器端分页等技术。

这几个原则在后面的几节中将分别讲解。

17.1.2　节约原则与 ViewState

若页面中不使用某对象，就应该将该对象关闭或者释放。本节将通过 ViewState 来举例说明节约原则。

【ViewState】

我们知道互联网使用了很多无状态的协议。用户在页面上的操作在大多数情况下服务器端是不知道的，这种限制对于实际 Web 应用造成了诸多不便。为了更好地保存用户状态，微软公司在 ASP.NET 中推出了 ViewState 技术，通俗地说，是在页面中加入了一系列隐藏

域（Hidden Field）来保存数据。不过该项技术会额外增加网络传输的成本开销，毕竟数据中包含了更多的状态信息，因此需要发送和返回的字节数都增加了。

节约原则应用之一就是对于 ViewState 的处理。每个页面控件都有 ViewState 这一属性。如果不会使用到，则完全可以将该控件的 ViewState 设置为 false，即禁用。而且，如果整个页面并不需要，完全可以将页面的 ViewState 都关闭。

【ViewState 的级别】

ViewState 实际上有 4 个级别：机器（machine）、应用（application）、页面（Page）和控件（Control），分别代表了从大到小的应用范围。机器与应用 ViewState 是通过配置文件来控制的，而页面和控件 ViewState 则可以通过在 Visual Studio 等 IDE 中修改属性值来实现。由于绝大多数基于 ASP.NET 开发的 Web 应用都会在某些页面采用 ViewState 技术，因此，完全关闭机器和应用 ViewState 并不现实。

如图 17-1 显示了在 Visual Studio 2008 中关闭控件 ViewState 的方法。

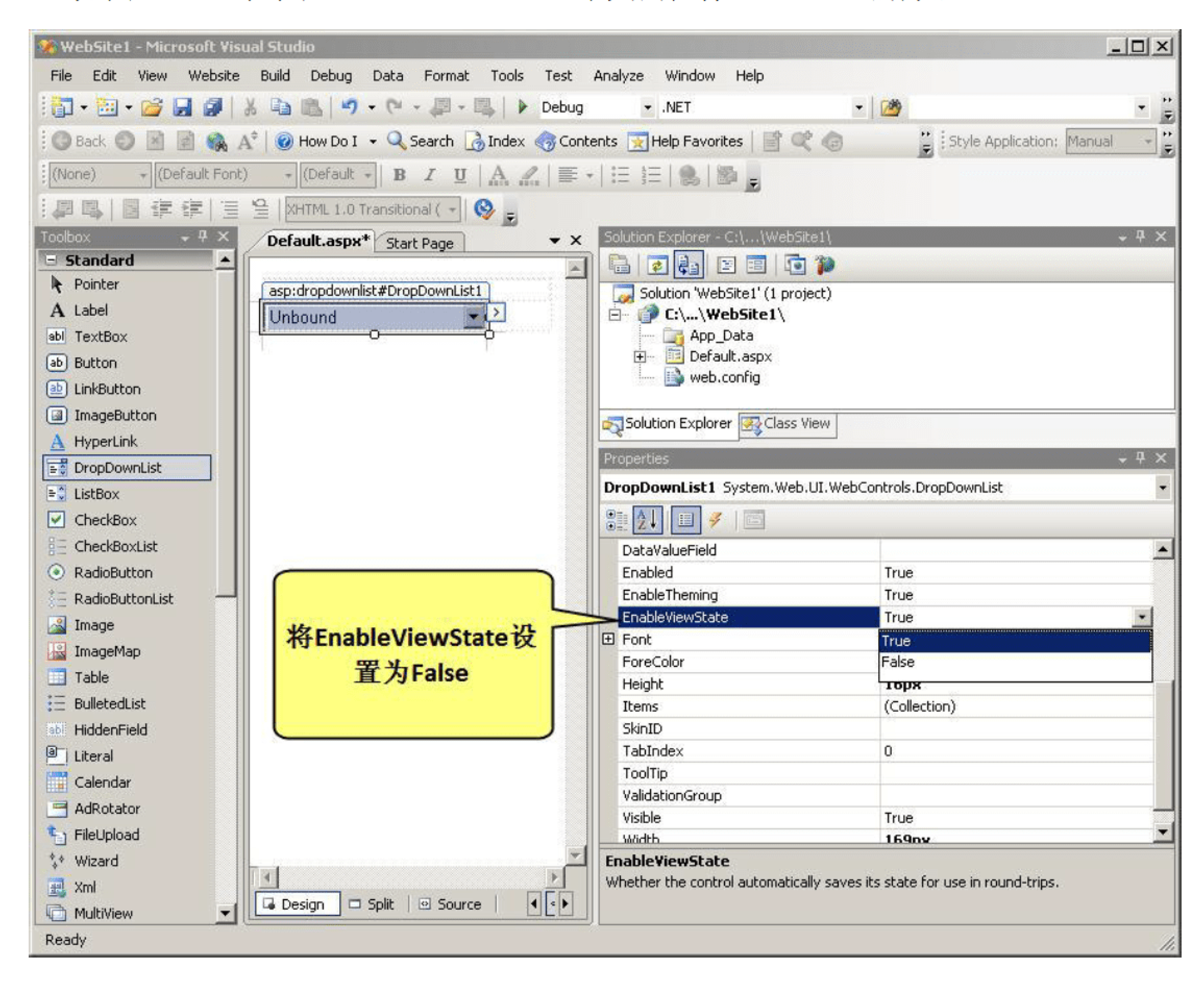


图 17-1　在 Visual Studio 2008 中关闭 ViewState

服务器控件的状态视图属性能够自动地在页面往返过程中维护服务器控件的状态，减少开发者的工作量，但是需要占用大量的服务器内存资源。因此，在不需要服务器控件状态视图的情况下，应将其 EnableViewState 属性设置为 false。

节约原则同样也适用于服务器控件的选择。

17.1.3　服务器控件的优化选择

在 ASP.NET 中，服务器控件带来的方便和功能是 html 控件所不能比拟的。但是每一个服务器控件都需要在服务器端创建相应的对象，是以牺牲服务器端的资源为代价的，过多地使用服务器控件会极大地影响程序性能。

很多情况下，简单地使用 html 标记或数据绑定即能够实现所需功能。比如<asp:Label>控件，若使用它来显示静态信息，则完全可用简单的标记来实现。如果 html 控件达不到所要实现的功能，而且在脚本语言如 JavaScript、VBScript 也不能实现的情况下，才考虑选择服务器控件。

17.1.4　恰当原则与 Session

除了使用页面上的 ViewState 来保存状态之外，ASP.NET（以及其他类似的 Web 开发技术）还提供了很多对象供开发者使用。在下面的文章中我们简单地以 Session 对象，服务器端控件举例说明这个原则。

【Session 对象】

Session 是什么呢？简单来说就是服务器给客户端的一个编号。当一台 WWW 服务器运行时，可能有若干个用户正在浏览这台服务器上的网站。当每个用户首次与这台 WWW 服务器建立连接时，他就与这个服务器建立了一个 Session，同时服务器会自动为其分配一个 SessionID，用以标识这个用户的唯一身份。这个 SessionID 是由 WWW 服务器随机产生的一个由 24 个字符组成的字符串。我们在前面的章节中已经接触到使用 LoadRunner 的参数来规避 Session 干扰的方法。

恰当原则应用于 Session 对象的合理应用。

我们知道，HTTP 协议是一种无状态的通信协议，无法记录和识别来自不同客户端的请求，但在实际应用中系统却要维护来自客户端的不同请求之间的会话状态信息。ASP.NET 通过将会话状态信息存储在进程、状态服务器或 SQL Server 数据库中来解决这个问题。

将会话状态信息保存在 Web 服务器的内存中具有最佳的性能，速度很快，但是却缺乏会话状态信息跨越多个服务器的能力。若要在多个 Web 服务器之间维护会话信息，可以使用状态服务器进行存储，这种方式由于可以将应用程序部署到多台服务器上而提高了系统的伸缩性和可靠性，但是以降低性能为代价。对于极其重要的会话信息，需要使用 SQL Server 存储方式，从而避免丢失重要的会话信息，但由此产生的工作负载比前面两者大得多。

若不考虑状态信息的保留和多个服务器共享，应尽量选择保存在服务器的进程中，从而得到最佳的性能。会话状态信息的存储方式可以通过修改 web.config 文件来实现，如代码 17-1 所示。

代码 17-1　在 web.config 文件中配置 Session

```
<sessionState
```

```
mode="InProc/StateServer/SqlServer" //不同的会话信息存储方式
stateConnectionString="tcpip=127.0.0.1:10000"
…
timeout="20"/> // 会话超时时间
```

17.1.5　Page.IsPostBack 的运用

ASP.NET 中还有一个很有意思的属性 Page.IsPostBack。它用于记录页面是否从客户端返回，若为 False 表示初次运行，否则表示从客户端再次返回该页面。Page.IsPostBack 的合理应用可以避免页面在往返过程中的一些不必要的操作。在 Page_Load()函数及一些只需要初始化一次的事件函数中均可以使用该属性来提高应用程序性能，如代码 17-2 所示。

代码 17-2　在代码中使用 PageIsPostBack

```
void Page Load(Object o, EventArgs e)
{
  if(! Page.IsPostBack)
  {
    conn=new SqlConnection("server=dataserver;uid=sa;pwd=;database=
    SchoolData");
    String sql="select * from teacher";          // SQL 查询语句
    cmd.Fill(ds,"Teachers");                     // 填充数据集
    mydataGrid.DataBind();                       // 绑定到数据表格
  }
…                                                //其他代码
}
```

以上代码将保证只有在首次访问该页面时，系统才对数据库进行读取并绑定，从而减少了之后对数据库的不必要读取。

17.1.6　合理使用 DataGrid 控件

对于采用 ASP.NET 的 Web 应用来说，DataGrid 控件一般是不可避免要采用的，因此需要特别列为一个小节来介绍合理使用它的规则。

DataGrid 控件本身带有最强大的数据显示功能，还内置了对数据的修改、删除、添加、分页等很多功能。但是，如果只需简单地显示数据，不使用其他更改功能的话，DataGrid 并不一定是最终和最佳的选择。DataGrid 控件的默认分页功能、对 ViewState 的依赖等，虽然能够让开发人员提高一些效率，但也会带来性能上的问题。

为此，ASP.NET 还提供了其他的一些灵活选择。比如 DataList 和 Repeater 控件。DataList 的功能比 DataGrid 要少，但却更加灵活，完全能够胜任 DataGrid 的数据显示功能。Repeater 控件很简单，但灵活性更强。这两个控件可以说是轻量级的 DataGrid，对于“轻量级”的页面非常适合。

另外，也可以采取传统的办法：对于复杂的页面，将数据显示和数据更新分开成几个页面，虽然这样增加了编写代码的成本，但增加了代码的可读性。在项目维护过程中，可以减少一些后期成本。所以，在需要显示数据时，选择 Repeater、DataList 还是 DataGrid 控件是需要实际考虑的。

17.1.7　合理进行字符串操作

在 Web 应用中经常会遇到字符串的操作，比如 Cookie、购物车相关的页面等。在 ASP.NET 中使用字符串（其他语言平台也具备类似的理念）需要注意如下几点：

1．使用值类型的ToString方法

在连接字符串时，脚本语言经常使用“+”或者“&”号直接将数字与字符串连接在一起，形成新的字符串。这种方法虽然写起来很简单也很直白，但在 ASP.NET 平台中是不建议这样做的。因为在 C#中，数字需要通过装箱操作（面向对象技术的一个术语）转化为引用类型才可以与原有字符串拼接。由于装箱操作涉及分配新对象、复制等操作，因此性能开销较大，当字符串处理较多时会影响 Web 应用性能。

避免出现此类问题的解决方法就是严格使用此类型的 ToString 方法。

2．运用StringBuilder类

String 类对象是不可改变的，对于 String 对象的重新赋值实际上是创建一个新的 String 对象再将新值赋予给它，因此性能开销较大。

使用 ASP.NET 在处理字符串时，最好使用.NET 命名空间 System.Text 中的 StringBuilder 类。该类对于字符串赋新值的处理有所不同，它通过各种方法（比如 Append、Remove、Insert 等）直接对字符串进行操作，并通过 ToString 方法返回操作结果，因此相比 String 会提高部分性能。

字符串操作是经常使用的基本操作，养成类似以上两点的编写代码习惯，会使得 Web 应用的性能有个良好的基础。否则，在广泛分布的多个页面中查找字符串带来的性能问题，将非常麻烦。

17.1.8　缓冲原则

所谓缓冲原则，是指在页面和应用服务器中多使用 Cache 这种技术。Cache 一般称为缓冲区，在港澳台地区也被音译为“快取”，顾名思义是指这样的一块区域中的数据能够相对较快地读取。

【Cache 浅说】

在计算机技术中引入 Cache 主要是为了更快地访问内存中的数据。CPU 中都有一定大小的 Cache 芯片，用于缓和寄存器与内存读写速度的巨大差异：内存中执行的一系列指令要处理的数据很可能也是连续的或者相同的，这样，把某些数据存放在 Cache 中，那么下一条指令就不必再去内存中寻找数据，而直接在 Cache 中调出即可；由于 Cache 速度比内存快很多，因此能提高整个计算机的性能。编程语言中的 Cache 也起到类似的作用，只不过它缓和了内存与硬盘读写速度的差异，使得硬盘上频繁访问的数据在内存中就可以找到，减少读写硬盘时间，从而提高应用程序的性能。

与很多技术平台一样，ASP.NET 也包含了 Cache 技术。它总共支持如下 3 种 Cache：

- 页面或控件 Cache；

- 应用程序级 Cache；
- 客户端浏览器的 Cache。

从上述 3 种 Cache 中也可以发现，实际上，ASP.NET 所支持的 Cache 无外乎两种：服务器端与客户端。前两种 Cache（页面、控件 Cache 与应用程序 Cache）均为应用服务器内存中的一块区域。客户端浏览器 Cache 则存在于客户端电脑硬盘中，一般是浏览器的临时文件夹或者登录用户的临时目录等处。在 Web 应用的持续运行中，随着内存的紧张程度不断增加，这些 Cache 均有可能失效，而其中内容均有可能在失效之前被提前删除。当每次用户请求一个页面的时候，浏览器会先从 Cache 里面去查找一下有没有符合要求的还没有过期的 Cache 内容，如果有的话就从 Cache 里面直接读取跳过网络传输。

在 Web 应用中增加 Cache 支持非常简单，下面按照前文所述的 3 个分类来介绍。

（1）页面或者控件 Cache 的声明和配置，在页面文件或者代码文件中编写都是可以的，比如代码 17-3 所示。为了解释清楚，依然采用了编程语言中的注释，请读者在实际编写代码时不要这么使用。

代码 17-3　在代码中声明使用 OutPutCache

```
<%@ OutputCache Duration="#ofseconds" //输出 Cache 的时间长度，单位为秒，必填项
   Location="Any | Client | Downstream | Server | None |
     ServerAndClient "           // Cache 的所在位置，默认为 Any
   Shared="True | False"         // 是否共享，默认不为多个页面共享，即 False
   VaryByControl="controlname"
                              //可以是逗号分隔的多个控件 ID，从而为列表中的控件提供缓存
   //可以根据不同的客户端定义字符串来区别建立 Cache，若为 browser 则可为不同的客户端
   //浏览器生成不同的 Cache
   VaryByCustom="browser | customstring"
   //可以根据 Header 不同而形成不同的 Cache
   VaryByHeader="headers"
   //可以根据 URL 中请求的参数不同形成不同的 Cache
   VaryByParam="parametername"
   CacheProfile="cache profile name | ''"
   // 下面是有关敏感信息的二级存储，即不允许把 cache 的内容（内存中）写到内存以外的其
   //他存储设备（比如硬盘，从而留下痕迹）上。在 MSDN 帮助文件中有更详细的说明
   NoStore="true | false"
   // 与数据库相关的设置，具体可查询 MSDN 帮助文件
   SqlDependency="database/table name pair | CommandNotification"
%>
```

在各个参数中，VaryByCustom 除了在上面的代码中声明之外，还需要在 Web 应用的 Global.ascx 中具体实现，如代码 17-4 所示的片段。

代码 17-4　在 Global.ascx 中声明使用 Custom string

```
Public override string GetVaryByCustomString ( HttpContext context,string
custom)
```

代码 17-4 中声明的字符串参数 custom 就是在代码 17-3 中 OutPutCache 定义中 VaryByCustom 的值。通过这样的设置，可以对不同的 custom 参数返回不同的字符串值以表示不同的 cache 内容。

（2）应用程序级 Cache。

应用程序级 Cache 最为自由，可以由开发人员自定义所需要的 Cache 功能。其只能在 ASP.NET 页面的代码文件中实现，具体做法是通过调用 System.Web.Caching.Cache 的各种方法，使用时可以通过调用 Page.Cache 来获得内容。

（3）客户端浏览器的 Cache。

客户端浏览器 Cache 与应用程序级 Cache 一样，也只能在代码文件中设置，具体方法是使用 Response.Cache，它属于 System.Web.HttpCachePolicy 类。

17.1.9　CLRProfiler 的安装与基本操作

对于利用.NET 平台开发的程序或 Web 应用，微软提供了一个可以免费下载的工具 CLRProfiler 来确认系统存在的性能问题。在本节中我们将学习如何使用 CLRProfiler。

1．CLRProfiler的下载与安装

CLRProfiler 有针对.NET 1.1 和 2.0 版本之分，后者的下载位置为：

http://www.microsoft.com/downloads/details.aspx?FamilyId=A362781C-3870-43BE-8926-862B40AA0CD0&displaylang=en。

（1）下载之后，选择安装目录解压缩后即可使用。安装后的文件夹结构如图 17-2 所示。在 Binaries 文件夹中分别有针对 x86 和 x64 操作系统的两个版本。

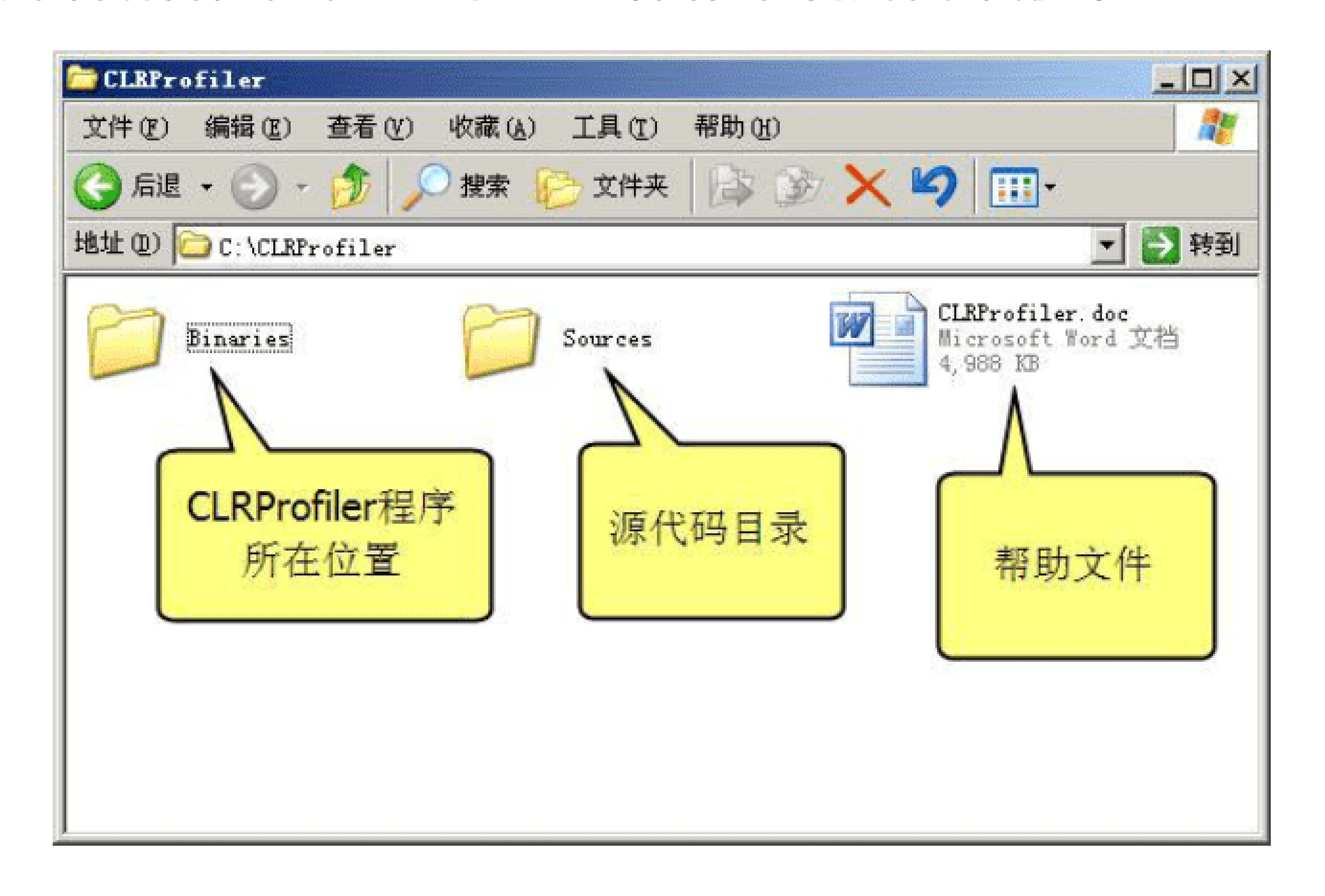


图 17-2　CLProfiler 安装后的文件夹结构

（2）双击 x86 文件夹中的程序图标运行 CLRProfiler.exe，程序界面如图 17-3 所示。

（3）在图 17-3 中 CLRProfiler 前面窗体上选择 File 选项（如图 17-4 所示），它很好地显示了 CLRProfiler 所能进行取样的各种程序、应用类型，在这里就不再赘述了。

2．CLRProfiler的使用

下面举例来说明 CLRProfiler 的用法。代码 17-5 是一个很简单的.NET 程序，它能够读取所在目录下名为 res.dat 的数据文件，统计其中的行数以及项目总个数。数据文件 res.dat 的格式如图 17-5 所示，它是由 SQL Server 导出工具所产生的按 Tab 键进行数据分隔的文本文件。

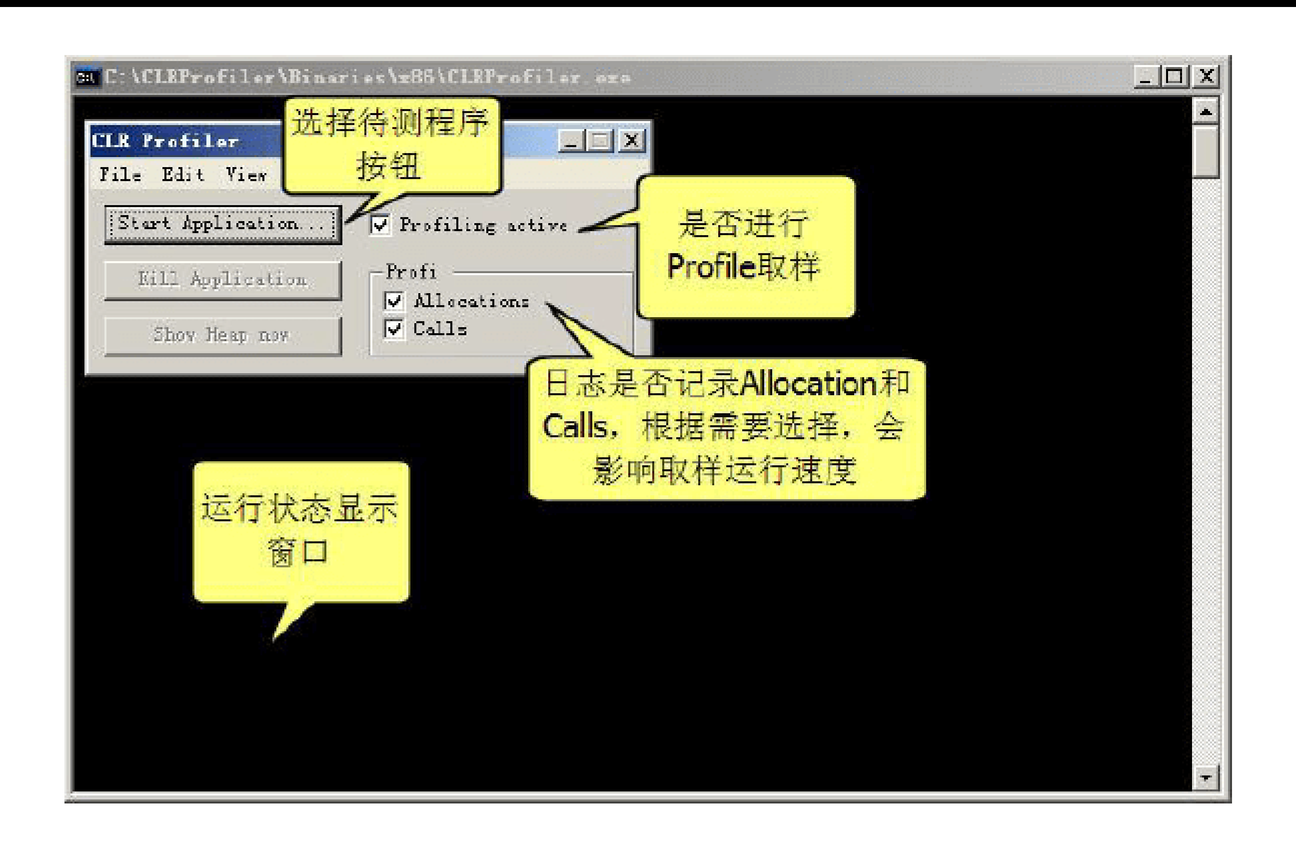


图 17-3　CLRProfiler 运行时界面

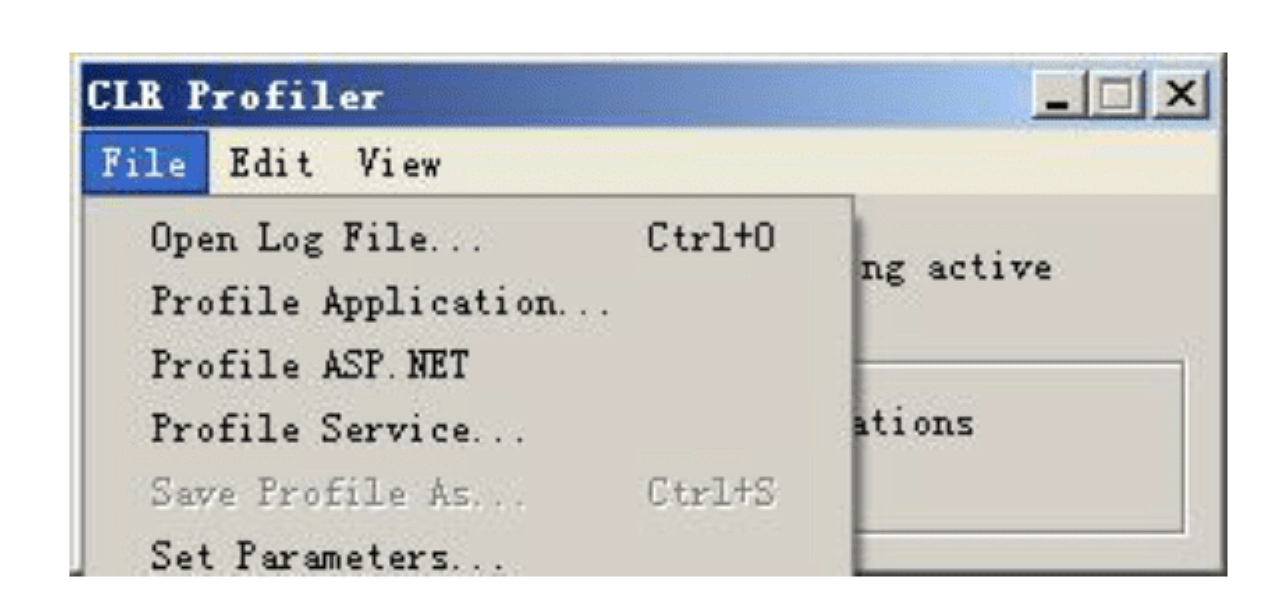


图 17-4　CLRProfiler 的文件菜单



图 17-5　数据文件 res.dat 的内容范例

代码 17-5　读取 res.dat 文件的简单.NET 程序

```
using System;
using System.IO;

class TestCLR
{
    public static void Main()
    {
        StreamReader r = new StreamReader("res.dat"); //待读取的文件
        //必要的初始化工作
        string strline;
        int lineNumber = 0;
        int itemNumber = 0;
        //读取文件，每次一行
        while ((strline = r.ReadLine()) != null)
        {
            lineNumber++;
```

```
            string[] items = strline.Split();         //拆分每行数据，获取 item
            for (int i = 0; i < items.Length; i++)
            {
                itemNumber++;                         //项目数加一
            }
        }
        r.Close();
        //控制台打印出总计读取的行数与项目数量
        Console.WriteLine("Read {0} lines, {1} items in total", lineNumber,
       itemNumber);
        //等待按键退出
        Console.ReadLine();
    }
}
```

单击 CLRProfiler 的 Start Application（开始程序）按钮，会提示一个打开文件的对话框。我们需要在其中找到代码 17-5 编译后的 exe 运行文件 testclr.exe。在选中文件之后，testclr.exe 开始运行，而 CLRProfiler 的 Kill Application（主界面中止程序）和 Show Heap now（现在显示堆）两个按钮也被激活，如图 17-6 所示。

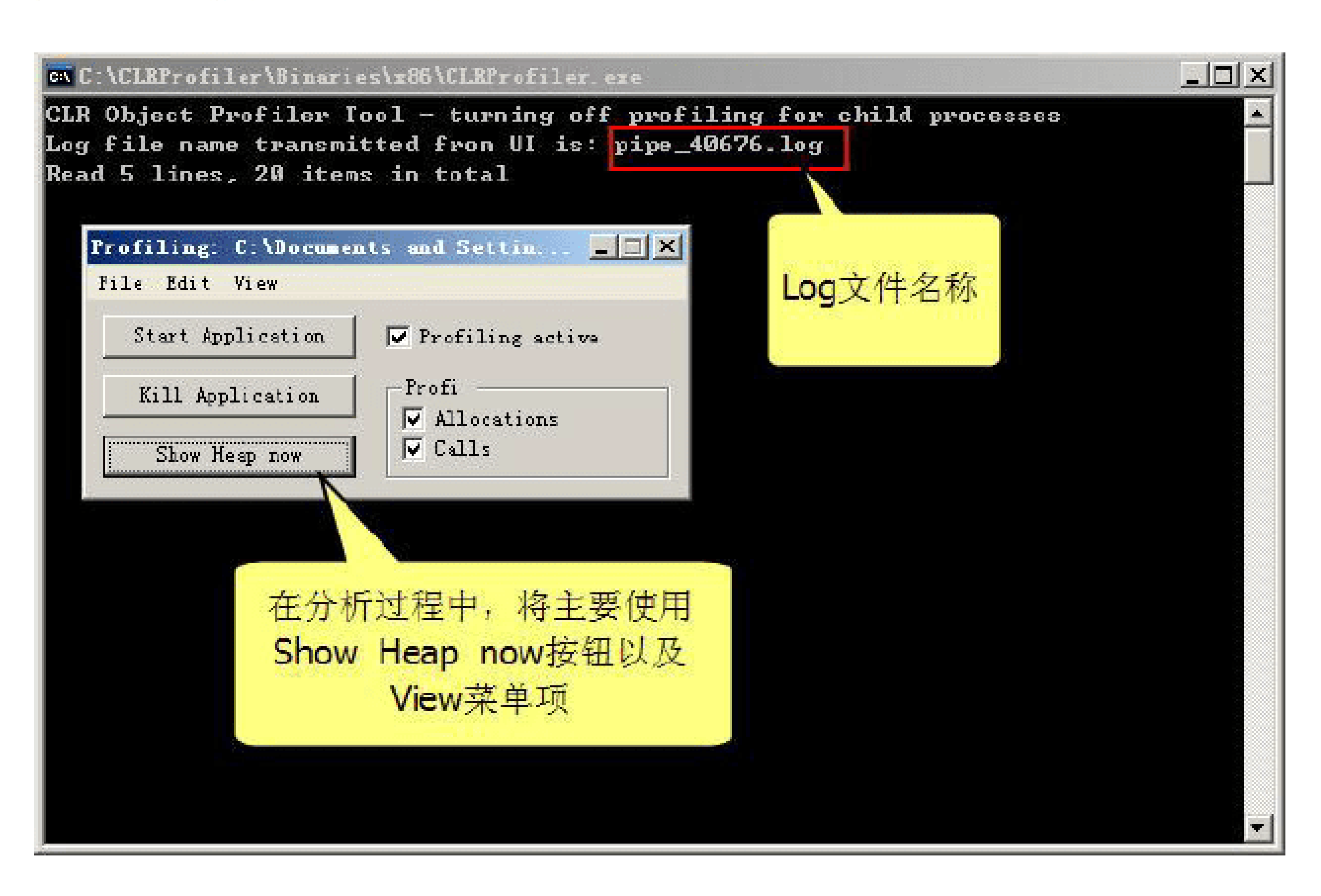


图 17-6　运行 testclr.exe 后 CLRProfiler 的界面

通过选择 CLRProfiler 界面上的 View 菜单以及 Show Heap now（现在显示堆）按钮，可以进行进程托管堆、垃圾回收机制的分析，能了解代码分配了多少内存，引起了多少垃圾回收和占用内存多久。举例来说，依次选择 View|Summary（查看|概要）命令，将显示本次运行的概要，如图 17-7 所示。

而在主界面中单击 Show Heap now（现在显示堆）按钮，可以查看当前堆的信息，如图 17-8 所示。

在图 17-8 中有很多方框构成的分支，在树的最末梢是对象。可以在图中空白处右击，将弹出图中所示的快捷菜单。其中 Show New Objects（显示新对象）选项将在 17.1.10 节提到，其对发现内存泄露有一些帮助。

【有关使用 CLRProfiler 的特别注意】

CLR Profiler 是一个插入式（Instruction tool）的工具，即在程序代码中增加若干指令以获得详细的信息，因此，它会使被检测的应用程序执行起来比起正常情况（即不使用该工具时）显著变慢。需要特别注意的就是，只能在测试环境下，而不要在生产环境中使用该工具。

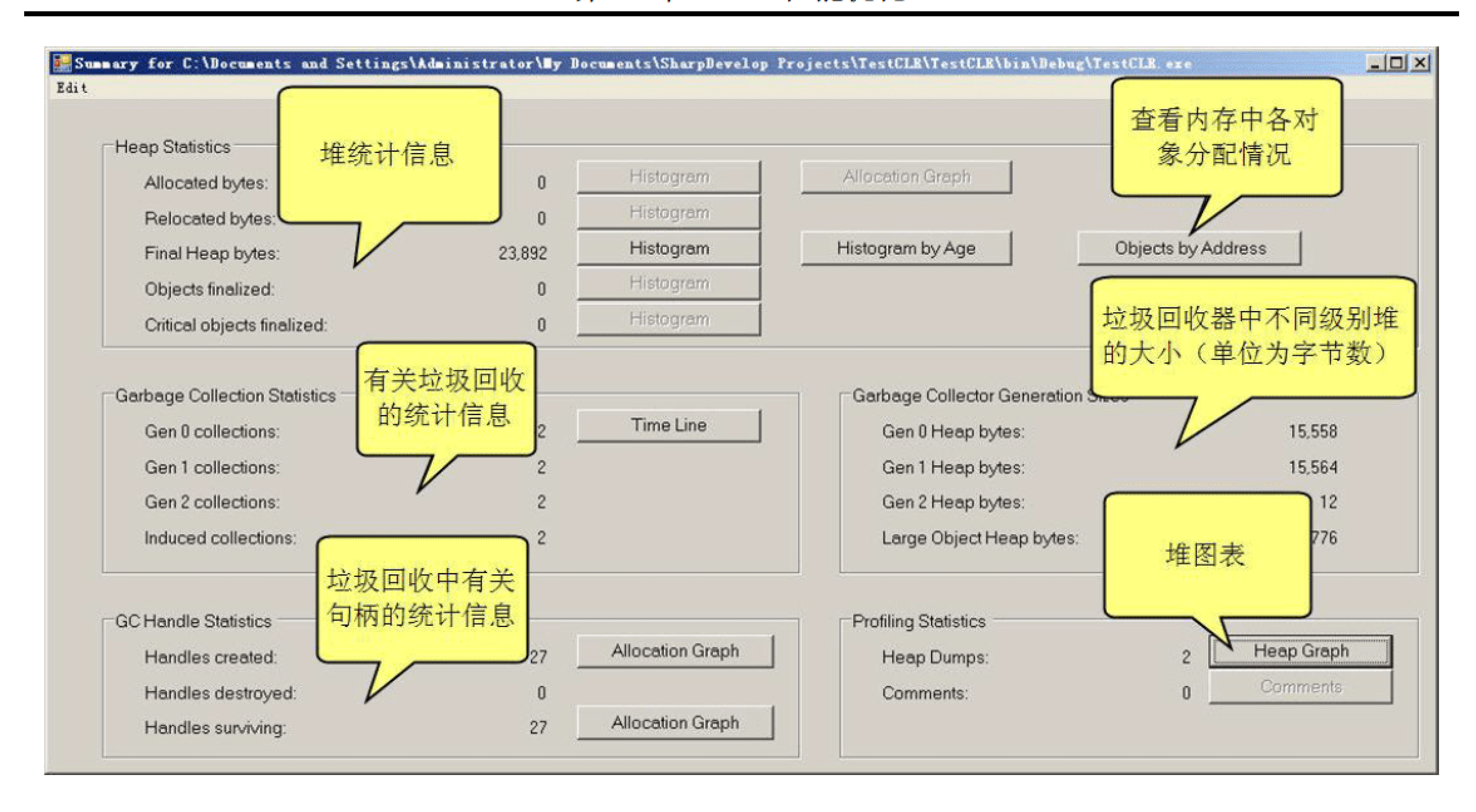


图 17-7　运行概要窗体

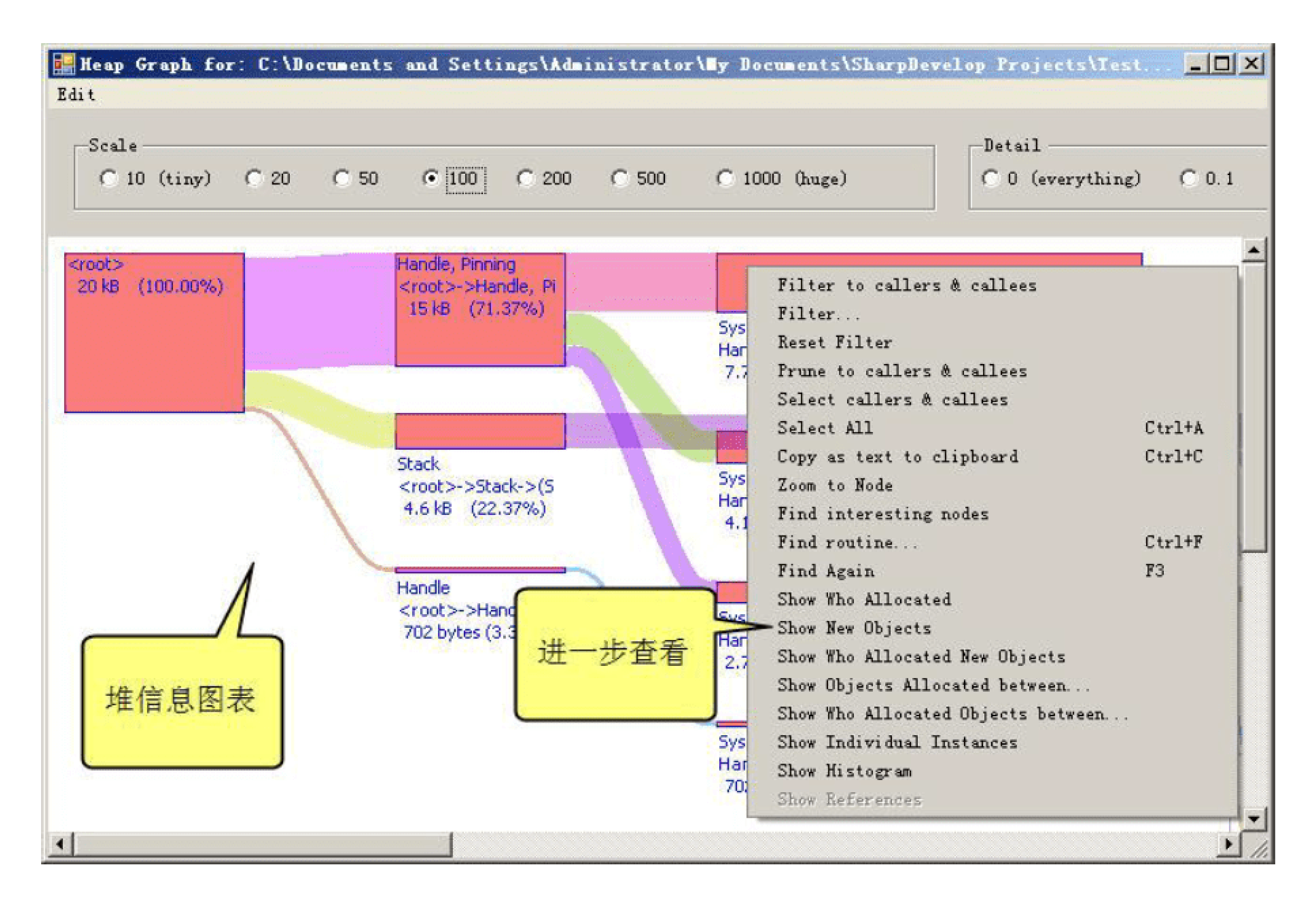


图 17-8　单击现在显示堆按钮能打开堆图表（Heap Graph）

代码 17-5 所列举的是一个传统的 Windows 程序，其实 CLRProfiler 同样可以对 ASP.NET 应用进行分析。方法很简单，只需要选择 File|Profile ASP.NET（文件|取样 ASP.NET）即可。但与图 17-6 有所不同的是，CLRProfiler 将首先自动重启 IIS，使用者单击启动程序（Start Application）之后，CLRProfiler 将弹出等待窗口，直到使用者通过 IE 浏览器打开本地存放的测试 ASP.NET 应用页面。

当使用者结束测试后，单击 Kill Application（停止程序）按钮，CLRProfiler 将清除运行时设置的环境，并再次重启 IIS，从而结束此次分析。

17.1.10 节我们将学习 CLRProfiler 如何进行内存的分析。

17.1.10　CLRProfiler 分析内存分配问题

CLRProfiler 的主要功能在于明了和发现在内存分配、垃圾回收方面应用可能会存在的信息或问题。比如：

- ❑ 过度的分配
- ❑ 未知的分配
- ❑ 内存的泄露

等。由于本书面向的是性能测试工程师，并非开发工程师，因此对于以上几点中的理论不会涉及，只通过分析一个简单程序的结果来显示获得信息的途径。

当被分析的程序运行完毕后，CLRProfiler 会生成日志，并弹出运行概要窗体，即类似前文所列出的图 17-7。在程序运行中，概要窗体也可以通过在界面中单击 Show Heap Now（现在显示堆）按钮来打开。另外，使用者还可以单击 CLRProfiler 界面的 View 菜单，通过其中若干子菜单项，来查看各种有关内存分配、垃圾回收的图表信息，如图 17-9 所示。

其实，View 中的各个菜单项在被分析程序运行的时候也可以查看。打开概要窗口，在 Heap Statistics（堆统计）一栏中 Allocated Bytes（分配字节数）后有个 Histogram（图表）按钮，单击后将显示出程序中各对象分配内存的大小情况。图 17-10 显示了某个字符串处理程序运行后的结果。

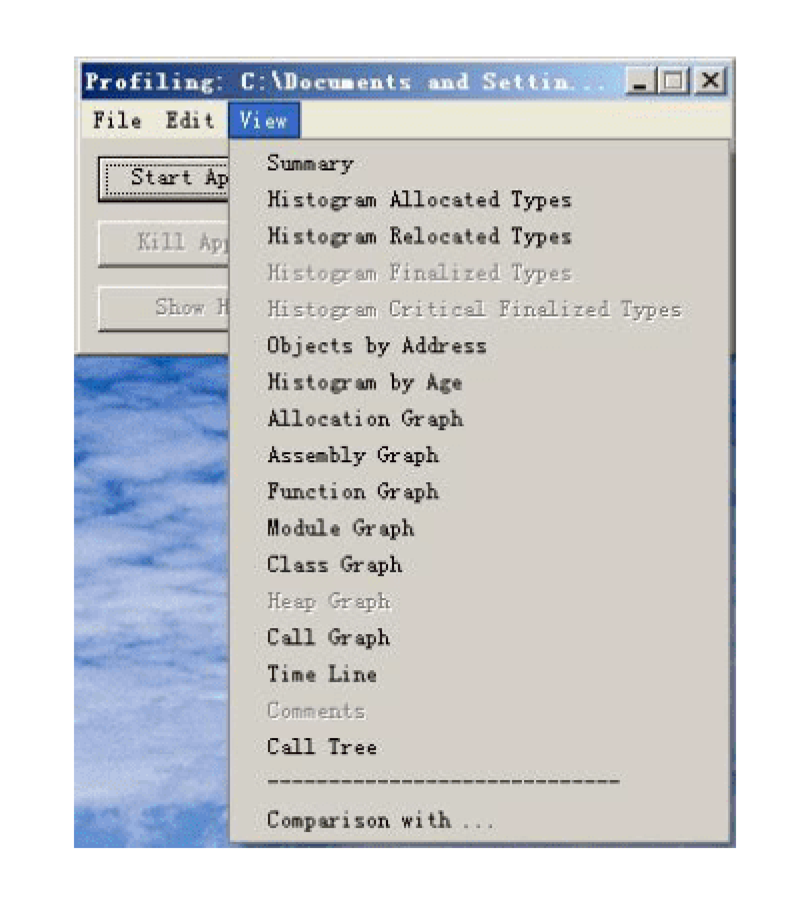


图 17-9　View 菜单中的各子菜单项

对象分配尺寸图指出了各对象类型在被分析应用的生命周期内被分配的情况。通过直观的柱图，可以发现哪些对象占用空间较大。

在 Heap Statistics（堆统计）一栏的第 2 项是 Reallocated Bytes（再分配字节数），其后也有 Histogram（图表）按钮。单击该按钮后显示的视图与图 17-10 很类似，但数值和幅度一般都较小，这是因为该图列出数值时对象依然被引用，因此垃圾回收机制会重新分配的结果。

通过前述两个图表数值上的比较，就可以发现在垃圾回收后，哪些对象依然存在。这样的对象一般称之为长命对象（Long-lived objects），当这些对象增多时，自然可用资源就会较少，影响应用性能。我们需要做的就是找到这些长命对象不合理存在的原因，并试图将其生命周期缩短。

前文提到，在 CLRProfiler 主界面单击 Show Heap Now（现在显示堆）按钮可以打开当前堆信息图表；现在，在该窗体内空白处右击并在弹出的快捷菜单中选择 Show New Objects（显示新对象）命令，即可查看 New Live Objects（新活跃对象）图表，如图 17-11 所示。该图表对于发现长命对象乃至内存泄露会有一些帮助。假设程序中有一段语句要循环 50 次，每次循环都新生成一个对象，那么总共 50 个新对象会被生成。如果图表中对应对象数目超过了 50 个，则需要特别注意。

以上简要说明了 CLRProfiler 使用以及分析上的一些要点，感兴趣的读者可以进一步查阅 CLRProfiler 的帮助文件。除了这个工具之外，其实还有更多的选择，但是对于.NET

或类似平台的分析，基本过程大致分为如下 4 步：

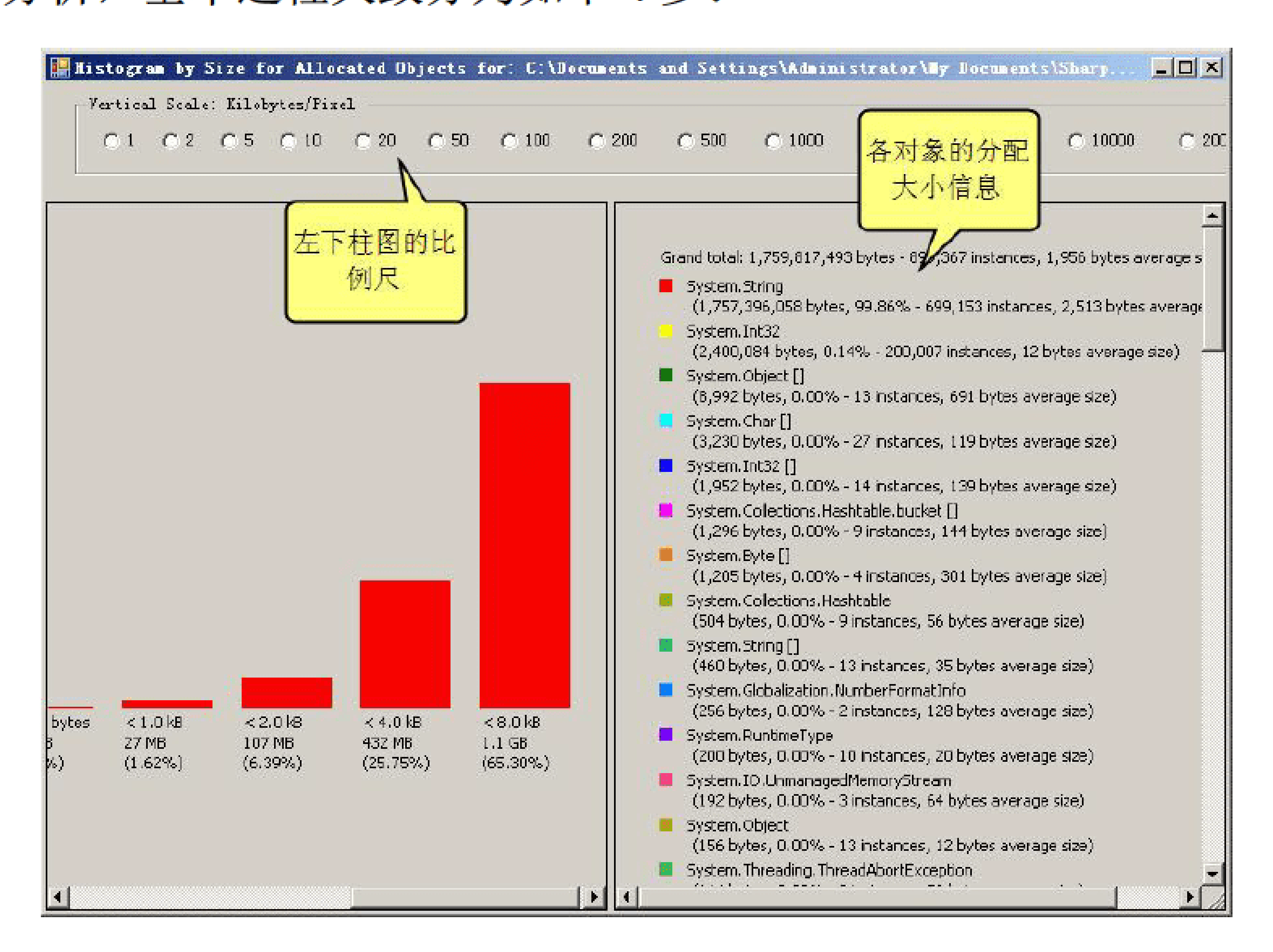


图 17-10　对象分配尺寸图

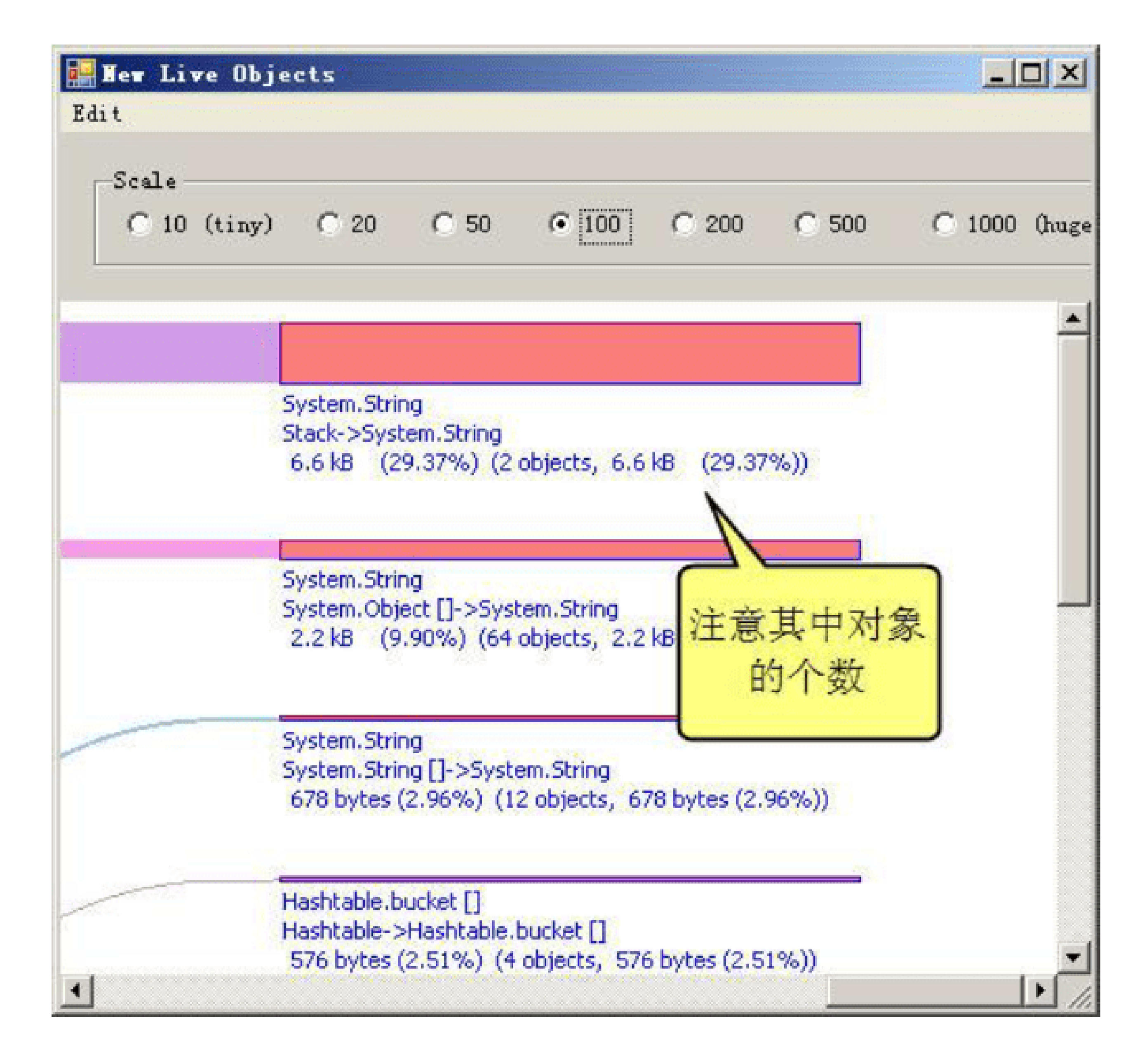


图 17-11　新活跃对象对象图表

（1）使用工具执行被分析程序。
（2）分析内存的分配类型。
（3）判断是谁分配了内存。
（4）分析哪些内存分配能够减少。

【测试工程师与内存分析】

在实际工作中，虽然以上这些任务主要是由开发工程师来完成的，但如果性能测试工程师能够有个基本的判断，对于快速发现和解决问题很有帮助，也能加深对程序的理解，更能促进团队的协作精神。

17.2　对应用服务器配置进行优化

网站的性能好坏取决于 3 个部分：软件、硬件及链接服务器和客户端的网络。对于软件性能，又可以分为两个子部分：Web 应用代码性能和 Web 应用所处环境的性能。同时，这几个部分并不是割裂的，相互之间可以互相影响和促进。

小白的公司网站采用 ASP.NET 进行开发，因此比较自然地使用了 IIS 进行网站管理。因此，在本节中，将以 IIS 性能优化的几个小技巧为实例介绍应用服务器配置优化。

17.2.1　启用 IIS 压缩

IIS 压缩就是为了提高应用服务器性能而采用的技术，因为它还可以减少网络的带宽占用。我们都知道下载一个大文件要比小文件时间花费的更长，而 Web 应用的大部分请求都是从服务器下载文件到本地，所以当下载的文件尺寸变小时，Web 应用的响应时间也会相应变短，从而提高了性能表现。

【IIS 压缩的简单原理】

IIS 压缩实际上就是把 Web 应用中的某些文件进行了类似 WinZip 的处理，使其尺寸变小。但是，我们拿到一般的 WinZip 文件并无法直接查看，需要进行解压缩。而浏览器在获得压缩过的文件后并没有解压缩就能显示，这是因为浏览器本身支持了这样的功能。为了让服务器端也得知客户端浏览器支持压缩，浏览器在发送请求服务器端返回文件时在请求头 Accept-Encoding 属性中将值设置为 gzip, deflate，服务器端发现后，即可以返回压缩过的内容。方法类似，也是在服务器反馈信息的头部 Content-Encoding 属性值设置为 gzip 或者将 Content-Type 属性值设置为 application/x-gzip。通过对内容做压缩标记，服务器和客户端的一来一往，就能够实现压缩数据的传递。绝大部分的浏览器（比如 IE 5.5 以上版本）都是支持数据压缩的。

按照下列几个步骤，就可以在 IIS 管理器的操作界面中启用压缩操作。在这里，本书以最常见的 IIS6 为例进行讲解。

（1）在服务器上依次选择“开始”|“所有程序”|“管理工具”|“IIS 管理器”命令，在弹出的窗体界面后右击左边列表框中的“网站”节点，选择 Properties（属性）命令。再在弹出窗体中选择 Service（服务）选项卡，就可以设置 HTTP 压缩中的若干选项，如图 17-12 所示。

（2）除此之外，为了压缩能够更好地工作，我们需要修改 IIS 元数据库中的某些属性，元数据库即 Metabase.xml 文件，记录了 IIS 的配置情况，默认是不允许修改的，因此需要 IIS 赋予编辑权限。通过在图 17-13 中单击授权选择框，就可以实现这个目标。Metabase.xml 处于%SystemRoot%\system32\inetsrv\目录下。

（3）在设置好 Metabase.xml 编辑权限后，就可以通过直接修改该文件来实现 IIS 压缩的细节了。通过记事本打开位于%SystemRoot%\system32\inetsrv\目录下的 Metabase.xml，找到 IISCompressionScheme 项，设置相关属性值，如代码 17-6 所示。为了讲解和阅读的方便，本书采用了类似编程语言的注释在 XML 代码中标出，在实际应用中这是不符合语法的。



图 17-12　在 IIS 6 中启用静态动态内容压缩



图 17-13　在 IIS 中启用 Metabase.xml 直接编辑权限

代码 17-6　修改 Metabase.xml 以设置压缩比

```
<IIsCompressionScheme  Location ="/LM/W3SVC/Filters/Compression/deflate"
        HcCompressionDll="%windir%\system32\inetsrv\gzip.dll"
                                                    //压缩算法库的位置
        HcCreateFlags="0"
        HcDoDynamicCompression="TRUE"               //是否开启动态内容压缩
        HcDoOnDemandCompression="TRUE"
        HcDoStaticCompression="FALSE"               //是否开启静态内容压缩
        HcDynamicCompressionLevel="0"               //设置压缩比
        HcFileExtensions="htm
            html
            txt"
        HcOnDemandCompLevel="10"
        HcPriority="1"
        HcScriptFileExtensions="asp
            exe"
    >
</IIsCompressionScheme>
<IIsCompressionScheme  Location ="/LM/W3SVC/Filters/Compression/gzip"
        HcCompressionDll="%windir%\system32\inetsrv\gzip.dll"
        HcCreateFlags="1"
        HcDoDynamicCompression="TRUE"
        HcDoOnDemandCompression="TRUE"
        HcDoStaticCompression="TRUE"
        HcDynamicCompressionLevel="0"
        HcFileExtensions="htm
            html
            txt
            js
            css
            htc"                    //压缩所支持的文件后缀名列表
        HcOnDemandCompLevel="10"
        HcPriority="1"
        HcScriptFileExtensions="asp
            exe
            axd"                    //压缩所支持的脚本文件后缀名列表
    >
```

```
</IIsCompressionScheme>
<IIsCompressionSchemes Location
="/LM/W3SVC/Filters/Compression/Parameters"
        HcCacheControlHeader="max-age=86400"
        HcCompressionBufferSize="8192"
        HcCompressionDirectory="%windir%\IIS Temporary Compressed Files"
        HcDoDiskSpaceLimiting="FALSE"
        HcDoDynamicCompression="FALSE"
        HcDoOnDemandCompression="TRUE"
        HcDoStaticCompression="FALSE"
        HcExpiresHeader="Wed, 01 Jan 1997 12:00:00 GMT"
        HcFilesDeletedPerDiskFree="256"
        HcIoBufferSize="8192"
        HcMaxDiskSpaceUsage="100000000"
        HcMaxQueueLength="1000"
        HcMinFileSizeForComp="1"
        HcNoCompressionForHttp10="TRUE"
        HcNoCompressionForProxies="TRUE"
        HcNoCompressionForRange="FALSE"
        HcSendCacheHeaders="FALSE"
    >
</IIsCompressionSchemes>
```

为了使得网站的动态 Aspx 文件也能够被压缩，在启动 IIS 动态内容压缩后，有必要将 aspx 这个后缀名加入到 HcScriptFileExtensions 这一属性数值中，这也是 Metabase.xml 需要手工修改的原因，在 IIS 管理器中并未开放实现此功能的修改界面。

通过前面所介绍的这几步操作，启用 IIS 压缩的设置已经完成。现在，我们已经可以发现压缩文件目录中出现了若干项，如图 17-14 所示。

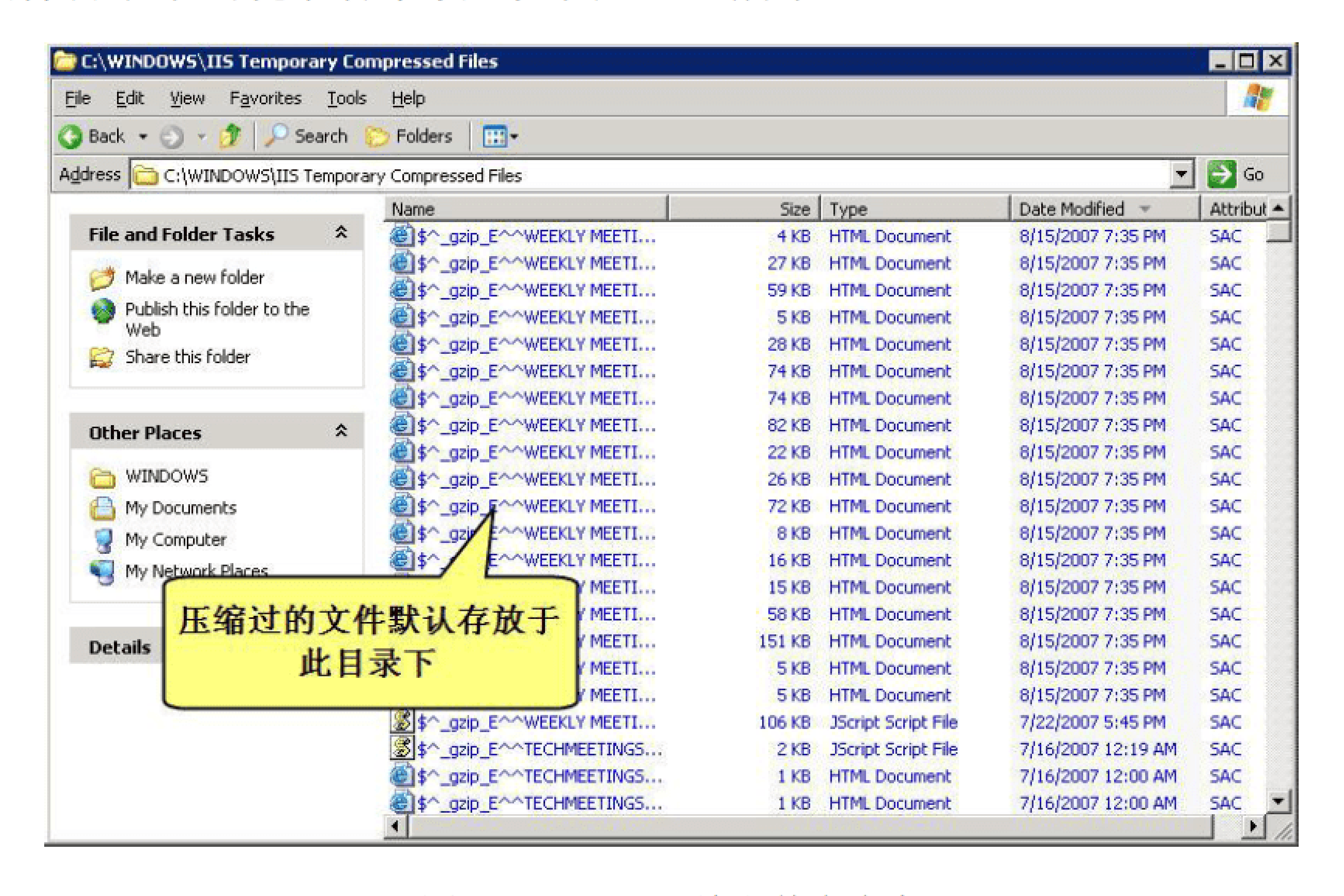


图 17-14　IIS 压缩文件夹内容

17.2.2　IIS 压缩比的选择

在前文中我们发现 Metabase.xml 中 HcDynamicCompressionLevel 属性的设置默认为 0 或者 10，这个数值被称为 IIS 的压缩比。如果这个值越大，则压缩比越大，处理后的数据

尺寸会越小。但是，大的数值会造成服务器负担增加，因此，在服务器性能与请求性能之间要有一个微妙的平衡。

根据 IIS 压缩比需要考虑平衡这一点，对于 Web 应用中的不同文件类型，有必要采取不同的压缩级别数值。具体的原则如下：

（1）对于一个请求大部分是图片类（jpg/gif/png）、某些文件类（pdf）的 Web 应用（比如网上相册、电子文档共享处理系统等），由于文件本身已经压缩处理过，再采用压缩比只会徒增服务器处理的负担，而不会提高性能。因此，在这样的应用下，压缩级别设置为 0 是可以的。

（2）对于大多数页面都是静态（HTML/HTM）的 Web 应用，可以采用较高的压缩级别，压缩过的这些页面在本地将储存在%windir%\IIS Temporary Compressed Files 文件夹中。由于静态页面的内容是固定的，只有第一次压缩更需要服务器 CPU 的处理，而之后 IIS 只要发送压缩后的页面即可，具备一次性的特点。因此，可以将压缩级别设置为最高，占用短暂的高 CPU 等资源占用是可以接受的。

（3）对于动态页面的压缩比选择，方式有所不同。动态页面随传入参数或者请求参数的不同，返回的内容是不同的，因此仅在第一次请求时进行压缩是远远不够的。由于每次都要压缩，占用资源就具有持续性的特点，如果每次都使用高压缩比，对整体性能的影响将比较大。据前人的经验，一般将压缩比设置为 4～5 是比较合适的。

17.2.3　IIS 7 压缩的进一步完善

为了使得压缩操作不过分占用服务器资源，IIS 7 推出了一个新功能：CPU 的 roll-off 属性。当 CPU 使用率超过一定数值时，压缩操作将停止。而当 CPU 使用率降下来低于设定的阈值时，压缩会再重新开始。这样的功能是通过设置 staticCompressionEnableCpuUsage /staticCompressionDisableCpuUsage 和 dynamicCompressionEnableCpuUsage/dynamicCompressionDisableCpuUsage 这两对属性来完成的。

【测试实战：在 IIS 7 中开启 Roll-Off 功能】

图 17-15 显示了在 IIS 7.0 中开启 Roll-Off 功能的方法，即添加或者修改%SystemRoot%\system32\inetsrv\config\ApplicationHost.config 文件中 httpCompression 标签的上述属性值。

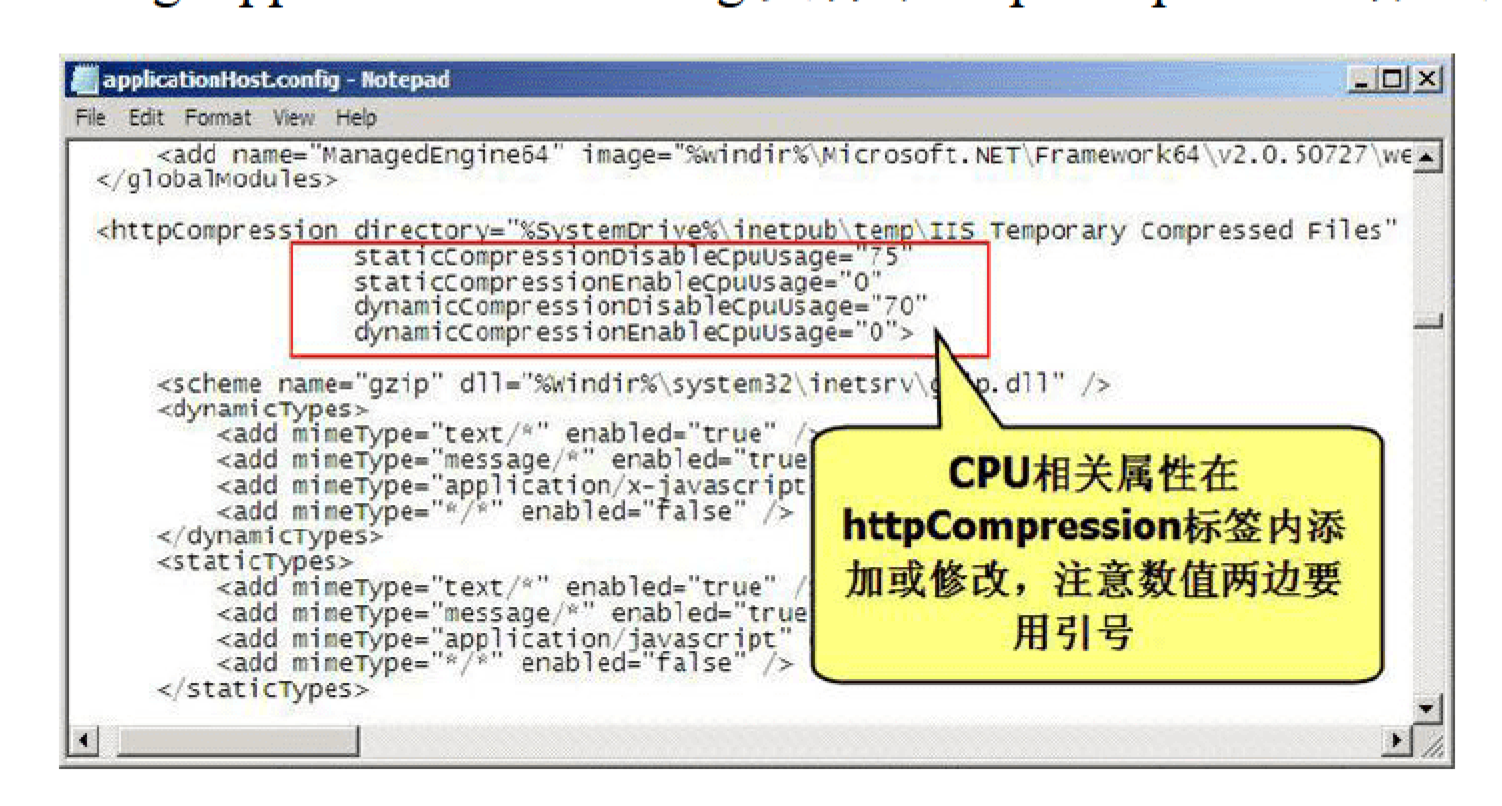


图 17-15　修改应用配置文件使得 IIS 7 支持压缩的 CPUrolloff 属性

对于这两对属性，IIS 7 的默认值如表 17-1 所示。我们可以根据实际情况进行相应修

改，一般会把不做压缩的上限进行调低处理，以保证系统的可用性。

表 17-1　IIS 7 中CPU rolloff相关属性的默认值

属　性　值	默认值（CPU 占用率，%）
STATICCOMPRESSIONENABLECPUUSAGE	50
STATICCOMPRESSIONDISABLECPUUSAGE	100
dynamicCompressionEnableCpuUsage	50
dynamicCompressionDisableCpuUsage	90

其实，IIS 6 同样也可以实现这样的功能，但需要单独购买第三方软件。

17.2.4　其他 IIS 性能优化措施

除了前面介绍的内容压缩之外，提高 IIS 的性能还有其他一些优化技巧。在本节中，我们将列举其中的若干。

1．打开IIS高速缓存

默认情况下，IIS 为所包含的服务提供 3M 的高速缓存。增大 IIS 高速缓存对于优化性能会有一定的帮助。前文讲过缓存的作用，IIS 高速缓存也起到同样的作用。Web 服务器会保留一部分内存空间，为 IIS 做高速缓存使用。这样，IIS 遇到将来的请求时，就可以从高速缓存中找到而不必要在硬盘中寻找，从而提高了响应速度。调整 IIS 高速缓存需要修改注册表，表项如下：

```
\HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\InetInfo\Parameters
```

在上述注册表分支中增加名为 MemoryCacheSize 的 DWord 键值，默认值为 3072000（3MB），可根据实际情况修改。如果该值设置为 0，则表示对请求不进行任何高速缓存。除此之外，还可以在上述位置增加名为 ObjectCacheTTL 的 DWORD 键值，用来指定高速缓存中对象过期时间（以秒为单位的十进制数字，默认为 30 秒）。设置完毕后重新启动 IIS 使得新值生效。

2．设置IIS连接属性

在 IIS 管理工具中可以对同时连接数、连接超时进行设置。当 Web 应用遭遇恶意攻击，限制同时连接可以起到一定的作用，使得网站减少崩溃的风险，提高资源利用的稳定性。连接的超时时间设置则可以使得 IIS 能够释放一些空闲过长的链接，减少资源的浪费。这两个属性的修改都是在管理工具界面“网站”节点处右击，在弹出窗体中分别选择 Performance（性能）和 Web Site（网站）这两个标签来实现，如图 17-16 和图 17-17 所示。其中，图 17-16 的上半部分还可以对带宽进行限制。

3．采取合适的SSL（安全套接层）加密策略

SSL 可以提供相当可靠的加密传输，但是由于其会带来一系列额外开销，将导致 IIS 服务器速度有所下降。因此，一般只对确实需要保护的目录进行 SSL 加密，另外还可以采

取修改 SSL 缓存保留时间的方法来提高性能：与本节前面所介绍的 IIS 高速缓存类似，在注册表中增加一个名为 ServerCacheTime 的 DWord 键值，修改 IIS 默认值 36000000（表示 10 个小时）为更合适的十进制数字即可。该键值应当加在如下位置：



图 17-16　限制同时连接数以及限制带宽

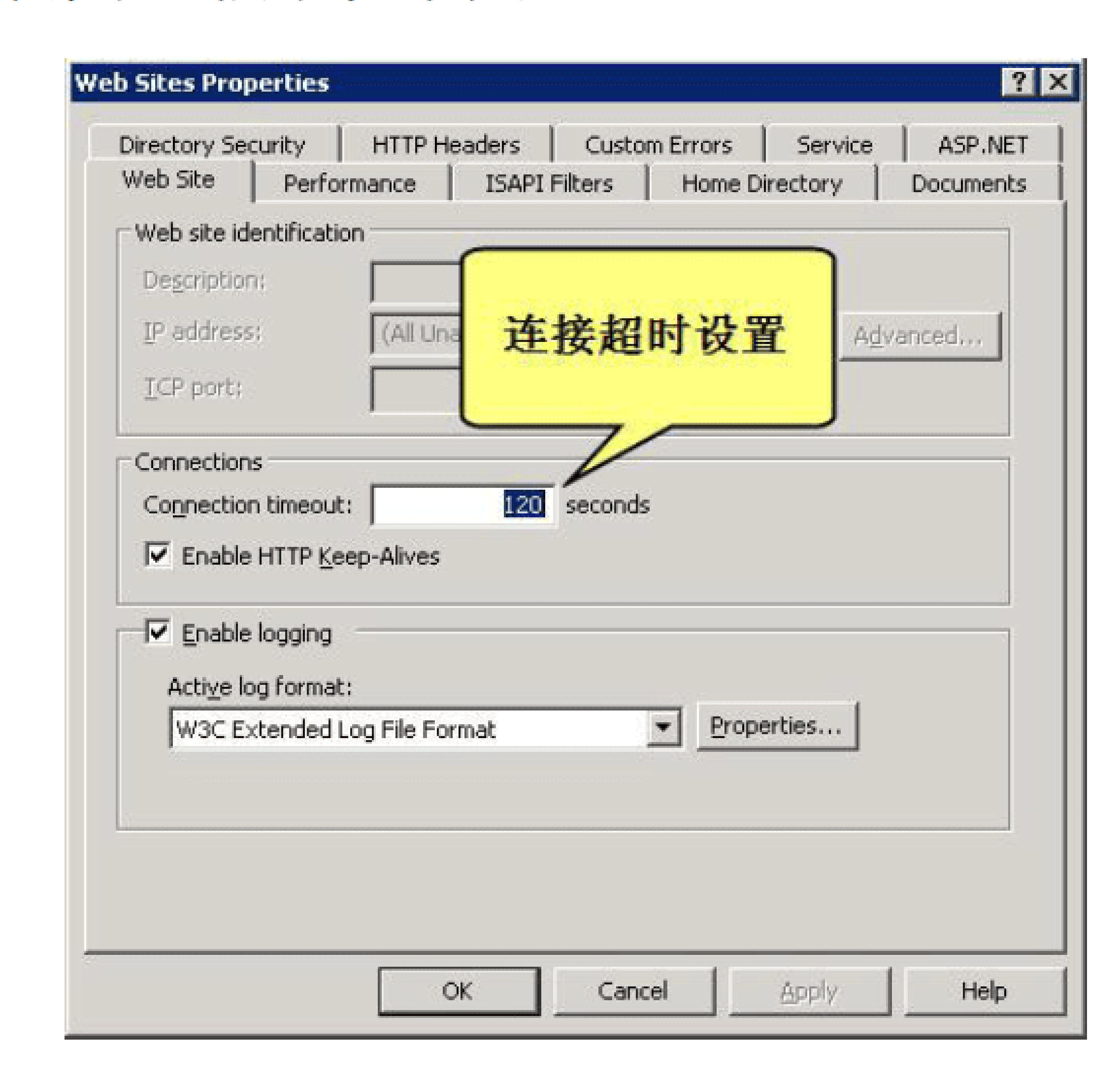


图 17-17　连接超时设置

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders
\SCHANNEL
```

4．优化本地网络连接更多地为IIS服务

在 IIS 所在服务器初次安装完毕之时，系统的文件缓冲区默认是为本地程序访问而优化的，对于 IIS 所处的网络环境并不很适合。因此，有必要进行修改。方法如下：

在 Windows 系统任务栏上依次选择“开始”|“所有程序”|“管理工具”|“网络连接”|“本地连接”命令，打开本地连接窗体并右击“本地连接”图标。在弹出的快捷菜单中选择“属性”选项，并在属性窗体中选择 File and Printer Sharing for Microsoft Networks（微软网络的文件和打印机共享选项）后，单击 Properties（属性）按钮，之后按照图 17-18 所示进行选择。

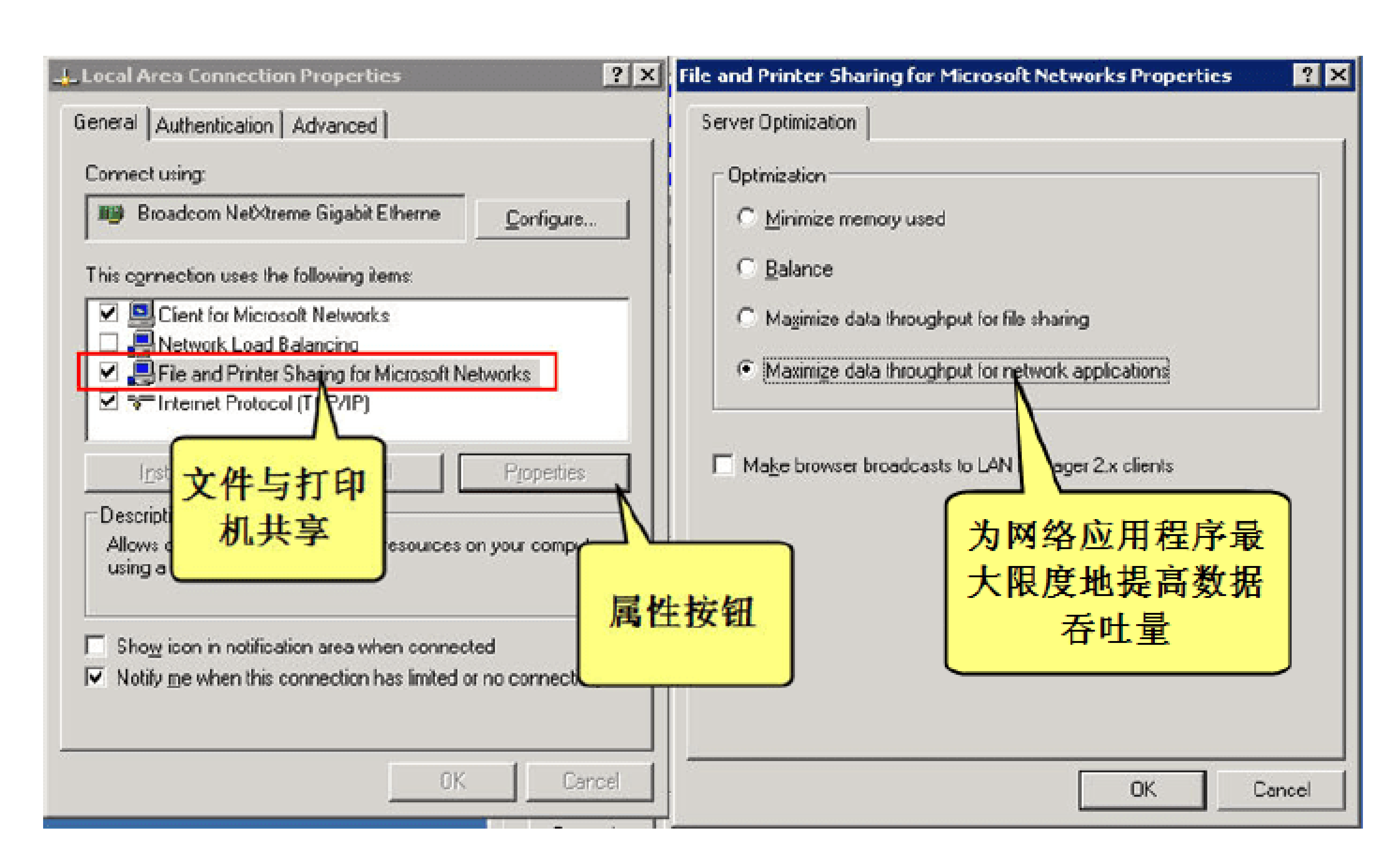


图 17-18　对本地网络连接进行优化

以上介绍了几个针对 IIS 的小优化技巧。虽然 IIS 具有安装简便、管理容易的特点，但是，真正理解 IIS 的运行机理，管理好 IIS 的各项设置并不容易，需要经验的积累和不断地学习。在 http://www.iis.net 上有很好的各种文章，介绍 IIS 的深入知识，读者可以延伸阅读。对于其他类型的应用服务器，在掌握了基本理念之后，具体的设置并没有太多根本的差别。

17.3　对数据库进行优化

数据库存放 Web 应用的数据。数据组织存储和被取出的方法，都会影响到整个 Web 应用的性能。数据库性能优化一般要考虑如下几个原则：

1．优化数据库的连接与关闭操作

Web 应用访问数据库一般都需要创建连接、打开连接、查询/更新数据这几个步骤，最后是关闭数据库连接这样的操作。如果每次查询、更新都建立一个新的连接，对资源的消耗则是巨大的。因此，Web 开发技术与数据库都提供了连接池（Connection Pool）的技术，能够将用户的数据库连接放在连接池中，当 Web 应用需要时可以直接使用，使用过后进入闲置状态，等待下一次的连接请求。当超过一定时间无请求，则会关闭连接以释放资源。在 Web 应用的代码中程序员也要注意及时关闭不再使用的数据库连接。

【使用连接池需要注意的问题】

连接池虽然提供了类似 Cache 的功能，但是毕竟大小是有一定限度的，如果 Web 应用代码在达到连接池上限后再继续请求创建新的连接，对于性能的影响会更加大：没有连接池的时候直接建立连接即可，现在需要查询连接池再建立连接，增加了步骤。所以，程序员需要注意的是在建立数据库连接后仅仅当真正需要数据库操作时才打开连接，使用完毕后尽可能地及时关闭，以减少数据库连接所占用的资源，提高连接池的可用性。

2．利用存储过程提高性能

所谓存储过程（Stored Procedure）是指存储在数据库服务器上的一组预编译的 SQL 语句。由于执行期间不必再编译，因此执行速度能有所提高。另外，存储过程在执行一次后，数据库引擎的执行计划一般会驻留在高速缓存中，在 Web 应用的其他代码中若调用同样的存储过程，则只需要利用该缓存中的二进制代码即可，会再次提高性能。

存储过程的好处还有就是节省网络带宽占用。在 Web 应用中，包含诸多条件查询的 SQL 语句在网络中传送会占用一定的网络带宽，但是用存储过程则会减少这方面的消耗。存储过程的名称相比查询语句较短，实现其内部逻辑的 SQL 查询语句在服务器端执行即可，而在 Web 应用代码中实现同样的逻辑则需要在服务器端和客户端进行多次 SQL 语句的传送，有关响应时间的性能会下降显著。

另外，存储过程存放于数据库中，独立于 Web 应用代码，便于提高安全性、便于维护修改，在需要升级、转移数据库时，更为方便。

3．使用优化的查询语句

Web 应用的代码无论多么简单，一般都会使用 SQL 查询语句进行数据库操作，存储过程也并不能解决所有的问题。因此，优化查询语句是非常重要的，也非常考验程序员的技术水平。因此，要尽量使用优化过的 SQL 查询语句以减少性能开销。具体的方法将在后面的内容中讲到。

性能测试工程师懂得 SQL 查询语言是很有必要的，因为不好的查询语言会使得其他方面优化很好的数据库前功尽弃，而且，查看 SQL 查询要比查看 C#等代码要相对容易一些，易于发现问题。

4．数据库的合理配置

数据库服务器运行时有很多的配置选项，可以对软件与硬件的配合进行适当的调整，以获得最优的性能。本节会在最后的部分简要介绍 SQL Server 的一些配置参数。

以上的前 3 个方面从软件的角度进行优化，第 4 个方面是从软硬件配合角度出发。除此之外，增加更快的硬盘、更多的内存等硬件升级方法也可以采用。但是，优化一般是指在现有条件下发挥更大功能而言的，纯硬件的升级提高的是服务器的整体性能，与运行于其上的数据库本身并没有太多关系，因此在本节纯硬件提高性能的方法就不再赘述了。

17.3.1　查询语句的优化

Web 应用的大部分数据库操作都是有关查询的。对于查询语句，要特别注意一些 SQL 语法关键词的用法。下面会列出一些常见的技巧。

1．In操作符与Not In操作符

In 关键词后一般是一个子查询语句，用括号包围起来。用 In 写出来的 SQL 语句具备自然语言的特性，比较容易读写、意义表达较清楚，因此很受开发人员青睐。但是，SQL 语句中包含 In 会导致性能有所降低。这是因为一般数据库引擎都会先试图将 In 语句转换为 Inner join 等多个表的连接语句：如果转换不成功再依次执行 In 后括号内、括号外的查询；如果转换成功则按照连接的执行计划进行。因此，使用 In 会增加一个转换的过程，导致耗费时间增加。如果在一个关系复杂的数据库中，这些转换所花费的时间甚至会数倍于不使用 In 关键字的时间。

至于 Not In，由于它不使用表上的索引，因此对性能造成很大影响，不建议使用。In 和 Not In 的替代关键字是 Exists 与 Not Exists。

2．对字段为空的判断与Null关键字

判断字段是否为空一般不会调用索引查询，而是进行全表扫描，这是因为数据库引擎所采用的 B 树索引并不索引空值(位图索引包含空值,但更多应用于联机分析处理 OLAP)。因此，根据实际应用情况，判断字段是否为空可以用完成相同功能的其他操作运算来代替，比如将 A is not null 改为 a>0 或 a>''等。另外，也可以用一个预先设定的特殊值来表示空值，这样会提高查询速度。

3. 字段值的比较：比较操作符（不等于、大于和小于）

比较操作符包括不等于（<>）、大于（>）和小于（<）等。

不等于操作符也不会用到索引，因此查询中包含它的话只会进行全表扫描。根据实际应用情况，可以用完成相同功能的其他操作运算来代替，比如 B<>0 改为 B>0 or B<0 等。

大于与小于操作符在执行 SQL 语句时会采用索引查找，但可以根据实际应用情况进行优化。比如：如果事先了解数据的分布规律，那么在查询时将条件写得越具体速度会越提高。

4. 通配符查询：LIKE操作符

LIKE操作符是Web应用代码中常见的SQL语句关键字，它可以应用通配符查询。虽然很方便，但是也要注意不好的查询语句所带来的性能问题。与上述的大于、小于操作符类似，书写查询语句时条件要尽量具体。比如对于查询第2位开始为ABC的字段，like '%ABC%'一般来说，就要比like '*ABC%'要花费更多的时间。

5. 联合查询：UNION操作符

UNION 操作符将多个表进行链接后会筛选掉重复的记录，因此会产生对结果集合的排序运算（要删除重复的记录以返回结果）。这样的操作导致数据量大的时候数据引擎不得不会利用磁盘空间来处理，从而性能变差。改善的方法就是采用 UNION ALL 操作符。它是简单地将两个表的查询结果合并后就返回，这样的处理在大多数 Web 应用中也是更为常见的。

总体而言，如下的 SQL 查询语句编写技巧是比较有用的：

- ❑ 尽量避免大块的事务操作，若可能将其分为多步进行，以提高整个 Web 应用系统的并发能力。
- ❑ 避免使用游标（Cursor），这是由于它的效率较差。可以考虑用临时表等来解决。
- ❑ 尽可能避免反复访问同一张或几张表，特别是数据量较大的表要避免在一个页面内反复访问。可以考虑的解决办法：根据条件提取数据到临时表中，然后再进一步操作；或采用把大表拆分的办法。
- ❑ 注意前文中列出的各种 SQL 查询语句编写问题，也要注意 where 后条件语句的编写。根据索引顺序、范围大小来确定条件子句的前后顺序，尽可能地让字段顺序与索引顺序相一致，范围从大到小。不要在 where 子句中的“=”左边进行函数、算术运算或其他表达式运算，否则系统将可能无法正确使用索引。
- ❑ 尽量使用 exists 代替 select count(1)来判断是否存在记录，count()函数只有在统计表中所有行数时使用，而且 count(1)比 count(*)更有效率。
- ❑ 注意表之间连接（Join）的数据类型，避免不同类型数据之间的连接。
- ❑ 注意插入（insert）、更新（update）语句可能操作的数据量，以防止与其他操作发生冲突。比如，若数据量超过一定的数据页面数，数据库系统将会升级锁，导致整个表暂时无法更新，使得别的操作无法进行。

【SQL Server 的索引优化向导很有用处】

为了提高 SQL 语句的执行效率、查询的效率，索引是必不可少的。对于一般用户，可

以通过建立包含查询的工作负荷文件，再调用 SQL Server 的优化向导来为各表建立合适的索引。

如果需要自己编写索引，那么请考虑如下原则：

- 索引的创建要与 Web 应用结合考虑，建议大的用于联机查询的表一般不要超过 6 个索引。
- 尽可能地使用索引字段作为查询条件，尤其是聚簇索引（clustered index）。
- 避免对大表查询时进行全表扫描（table scan），如果有必要可以考虑新建索引。
- 要注意索引的维护，设置周期性的索引重建计划，以适应数据量的变化。

17.3.2　查看 SQL 语句执行计划与数据库当前事件

17.3.1 节简要介绍了一些 SQL 查询语句关键字对于性能的影响。SQL 语句虽然相对简单，但是用好并不容易，而且好坏的差距很明显。对于 Web 应用中现有的 SQL，在可能的情况下，开发人员都要进行执行计划的分析，在性能测试之前进行所谓的“单元优化”。性能测试工程师也要了解如何查看 SQL 语句的执行计划。下面以 SQL Server 2005 为例，简单介绍操作过程。

SQL 语句执行计划是在 SQL Server 管理中心（Management Studio）中实现的，它可以通过依次单击“开始”|“所有程序”|“Microsoft SQL Server 2005”|“SQL Server 管理中心”命令打开。

在管理中心的界面，可以新建查询语句并执行它，如图 17-19 所示。默认情况下，查询语句的执行计划是不被显示的，我们可以通过图 17-20 所示的菜单将其开启。

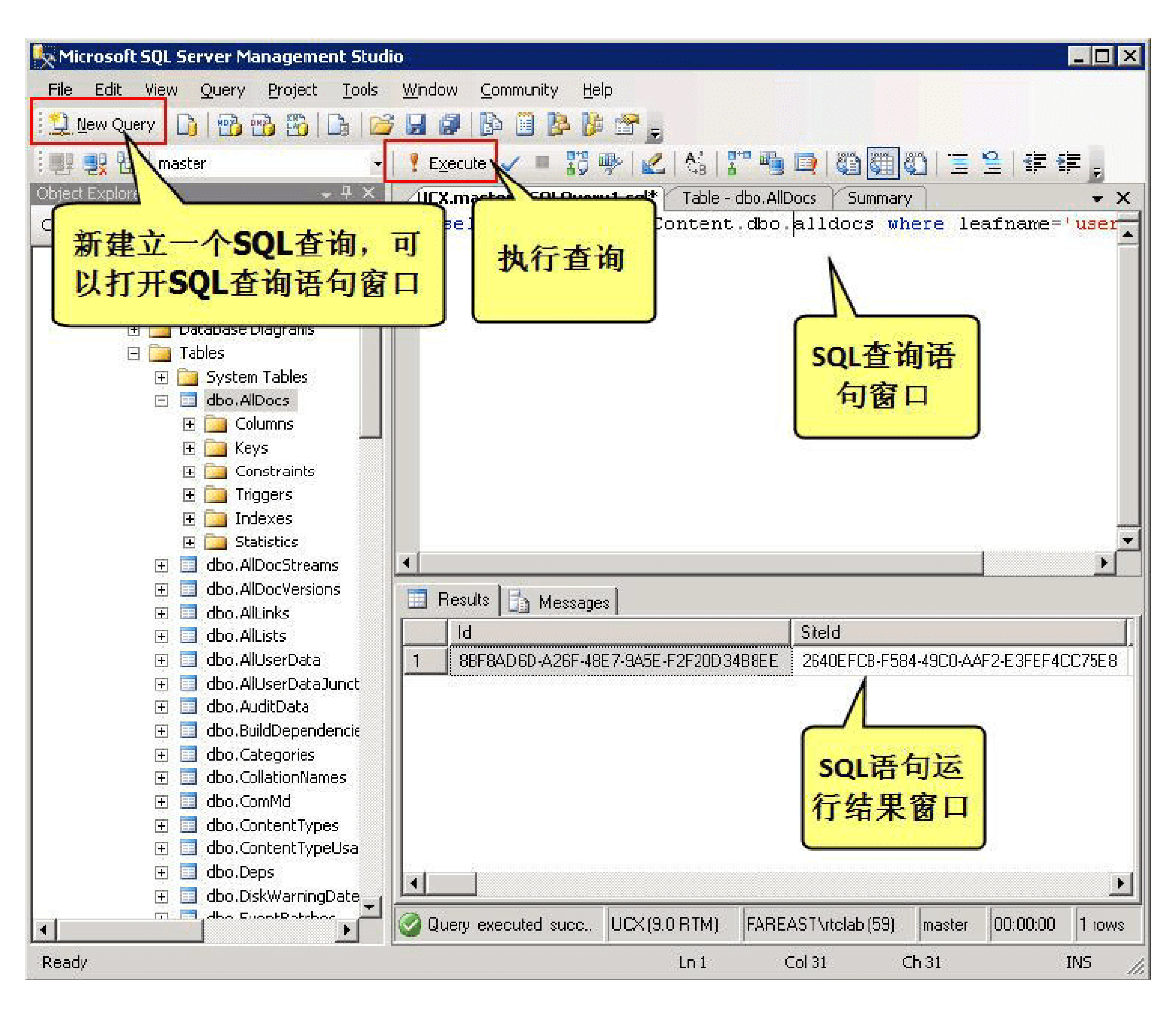


图 17-19　在 SQL Server 的管理中心编写并运行查询语句

除此之外，SQL Server 还提供了 Profiler 用来查看当前正在运行的各项事件。

在安装有 SQL Server 2005 客户端和管理工具的电脑上依次单击“开始”|“所有程序”

|Microsoft SQL Server 2005 命令，进一步依次选择其中的“性能工具”|SQL Server Profiler 命令。单击导航栏中的 New Trace（新跟踪）图标按钮，在弹出的身份验证窗体中输入相关信息后，即可连接到待分析的数据库服务器中，如图 17-21 所示。

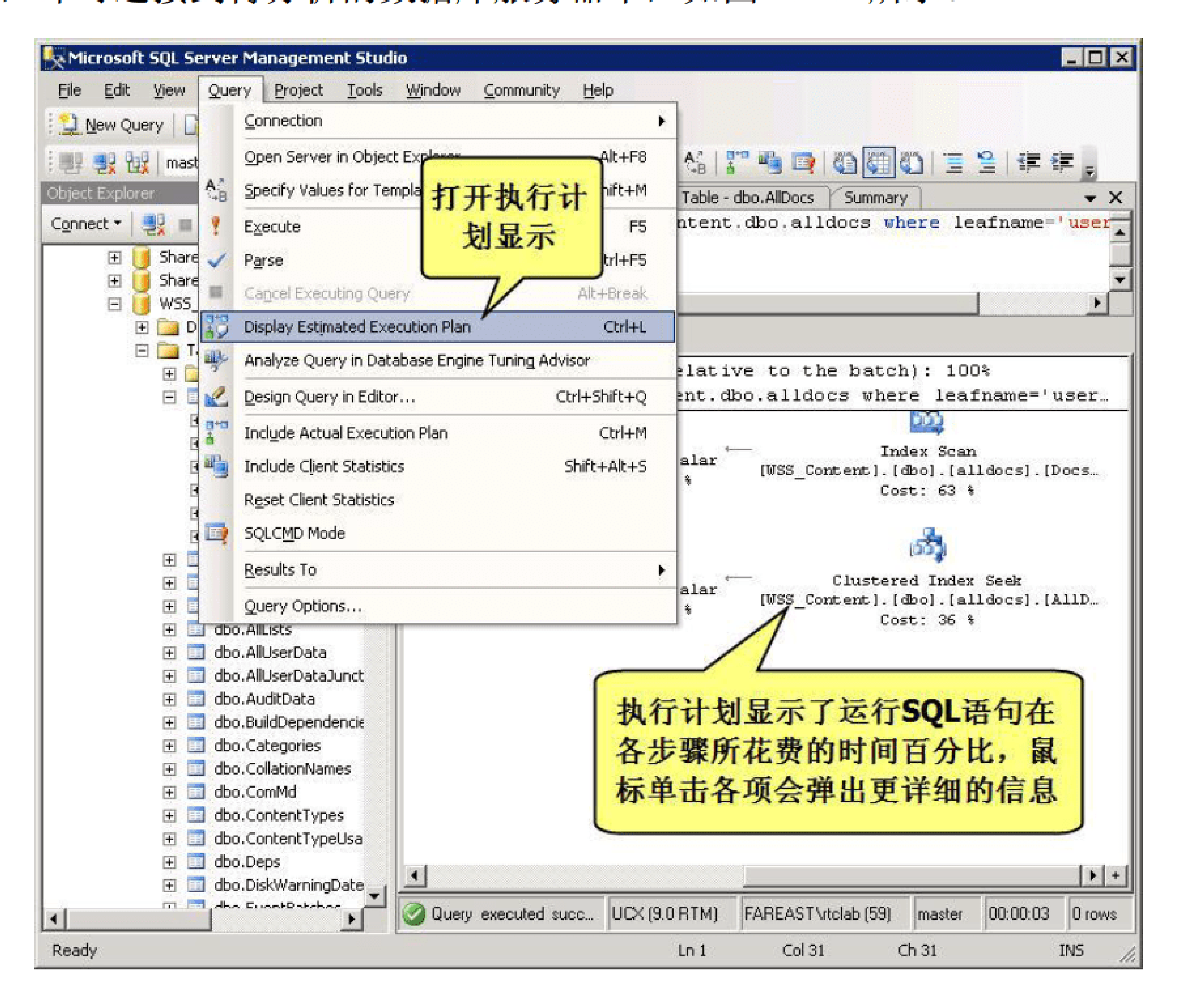


图 17-20　打开 SQL 语句的执行计划显示



图 17-21　在 SQL Server Profiler 中建立新跟踪

在图 17-21 中单击 Connect（连接）按钮，将弹出跟踪设置窗体，如图 17-22 所示。读者可以根据实际情况，对各项事件采取记录或者不记录的操作。同时，也可以单击右下方

的 Organize Columns（组织数据列）按钮增加需要跟踪记录的事件。设置完毕后，单击 Run（运行）按钮就可以开始跟踪了。

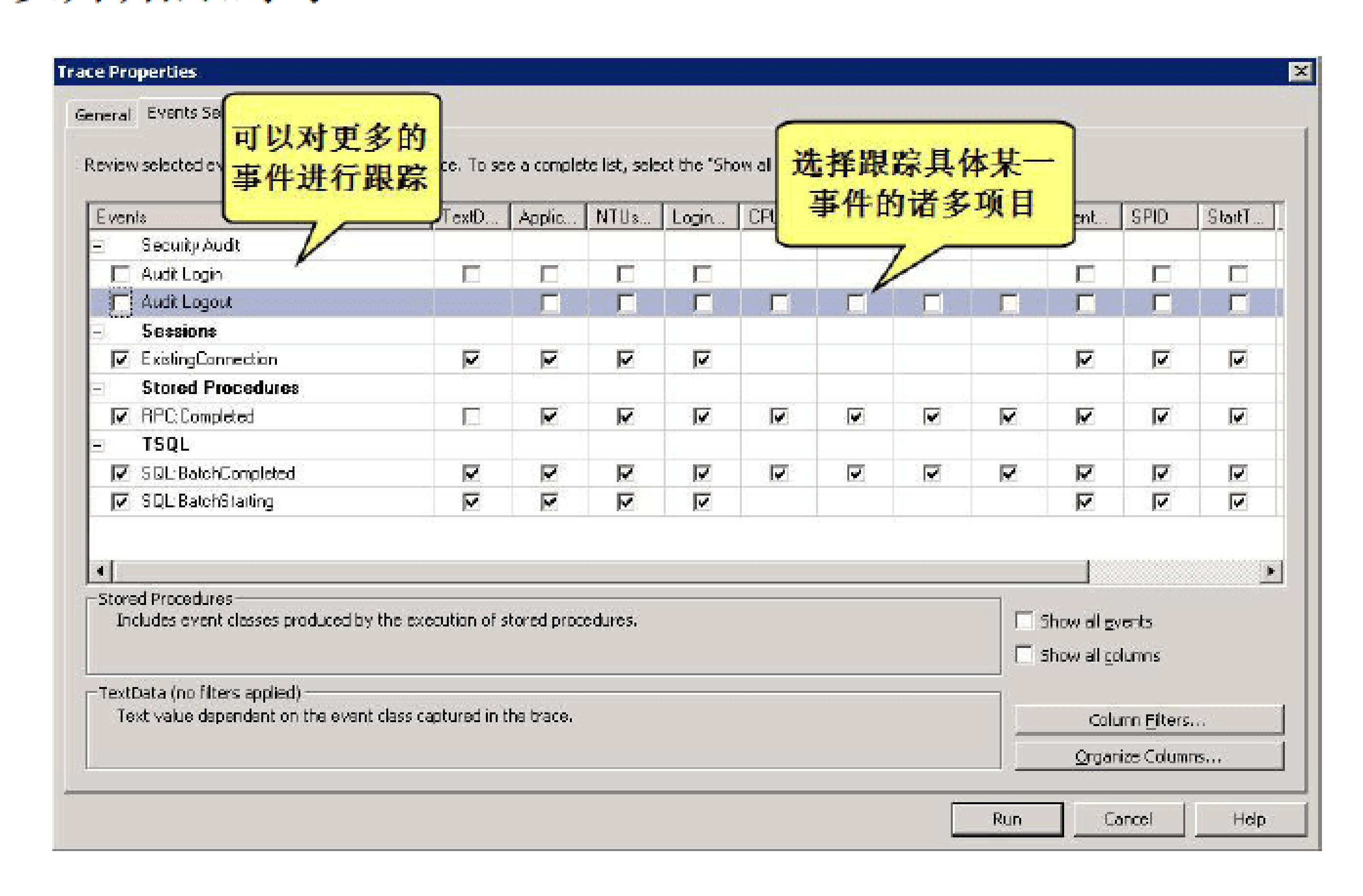


图 17-22　对跟踪的事件进行设定

SQL Server Profiler 运行时的界面如图 17-23 所示。单击每一行，在下方的窗口中均可以显示具体的 SQL 语句，非常方便。

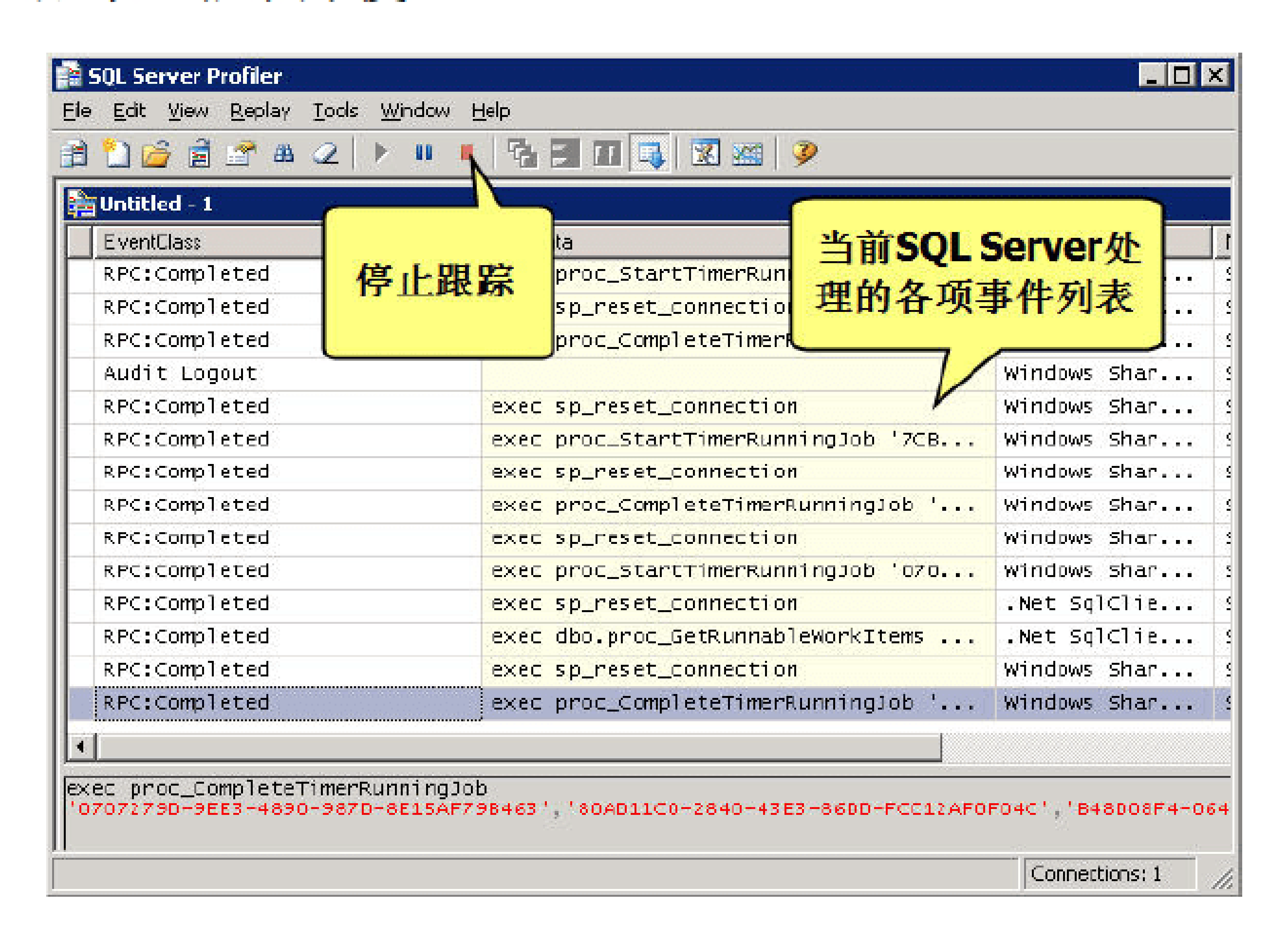


图 17-23　SQL Server Profiler 运行时界面

通过 Profiler，可以实时发现当前 SQL Server 正在进行的任务。当 Web 应用某个页面代码中的 SQL 较复杂、不容易阅读时，可以利用 Profiler 来获得实际的查询语句，对于调试很有用处。有关更多的 SQL Server Profiler 使用，请查阅相关的数据库书籍。

17.3.3　提高存储过程与自定义函数性能

在 Web 应用代码中使用存储过程和自定义函数，能够将业务逻辑封装在 SQL Server 内，使得页面与背后的逻辑分离得更好，易于维护，因此广为采用。

在具体的使用中，前人总结了如下的经验，需要在实施前仔细考虑：

（1）注意存储过程中参数和数据类型的关系，尽量避免数据类型转换开销。

（2）开发人员在提交存储过程之前，要通过前文所介绍的查看执行计划方法或者使用

set showplan on 进行分析，以确定查询是否可以进行优化。

（3）开发人员如果用到其他库的 Table 或 View，务必在当前库中建立 View 来实现跨库操作，最好不要直接使用“database.dbo.table_name”，因为 sp_depends 不能显示出该 SP 所使用的跨库 table 或 view，不方便校验。

（4）对于存储过程和自定义函数，要采用合理的算法、循环、字符串处理。

另外，对于存储过程中常用的临时表或 tempdb，也要注意：

- 尽量避免使用 distinct、order by、group by、having、join 等增加工作负担的语句。
- 避免频繁创建和删除临时表，减少系统表资源的消耗。
- 新建临时表时，如果一次性插入数据量很大，那么可以使用 select into 语句代替 create table 语句，以避免日志操作，提高查询速度。
- 如果所操作的数据量不大，为了缓和系统表的资源，建议先创建表(使用 create table 语句），然后进行插入数据（insert 语句）的操作。
- 如果临时表中的数据量较大，需要建立索引，那么应该将创建临时表和建立索引的过程放在单独一个子存储过程中，这样才能保证系统能够很好地使用到该临时表的索引。
- 对于临时表，在存储过程的最后务必将所有的临时表显式删除，先执行清理操作（使用 truncate table 语句），然后再删除表（使用 drop table 语句），这样可以避免系统表的较长时间锁定。
- 慎用大的临时表与其他大表的连接查询和修改，减低系统表负担，因为这种操作会在一条语句中多次使用 tempdb 的系统表。

17.3.4　数据库的硬件配置优化

数据库在安装完毕后，有时候需要根据实际情况进行启动和运行时的参数调整。对于 SQL Server 来说，通过企业管理器，或者在查询分析器中运行 sp_configure 存储过程，用户可以修改 62 个 SQL Server 配置选项（SQL Server 2005 版本的数目）。通过熟悉一个系统的配置调整之后，在遇到其他的数据库系统时，也比较容易上手，因为各个系统所面临、要考虑和解决的硬件环境问题是类似的。当然对于其他数据库系统，参考帮助文档中的参数配置、性能优化部分总是必须和非常有用的。

本节将简要介绍一下 SQL Server 的这些配置选项。为了更为清晰，将介绍运行 sp_configure 进行配置的方法。SQL Server 的 sp_configure 存储过程使用方法如代码 17-7 所示。

代码 17-7　sp_configure 的使用语法

```
sp configure [ [ @configname = ] 'option name'  [ , [ @configvalue = ]
'value' ] ]
//下面是一个例子。设置整个数据库的恢复间隔为 3 秒
USE master;
GO
EXEC sp_configure 'recovery interval', '3';
RECONFIGURE WITH OVERRIDE;          //使得新配置生效
```

对于 option_name，SQL Server 开放了如下的多个选项，它们也可以通过运行代码 17-8

获得列表。

代码 17-8　查询 sp_configure 中选项列表

```
SELECT * FROM sys.configurations
ORDER BY name ;
GO
```

在 SQL Server 2005 中运行如上语句，将查询到 62 个结果，即 option_name 的总数。

【启用高级属性修改】

在 SQL Server 2005 中有些高级属性是仅允许有经验的数据库管理员或认证的 SQL Server 技术人员进行更改，因此需要按照代码 17-7 的语法，首先执行 sp_configure 将 show advanced options 设为 1 才可以。至于具体哪些属性需要启用高级修改，可以根据运行 sp_configure 时的系统提示获得信息。

下面介绍一些与性能相关的 SQL Server 2005 可配置属性。

（1）ft crawl bandwidth：该属性用于设置 SQL Server 全文检索抓取内容时所用的带宽。最大值为 32767，最小值为 0，系统默认值为 100。

使用 ft crawl bandwidth 选项，可指定较大内存缓冲池能增长到多大范围。较大内存缓冲区的大小为 4 MB。

实际上，该属性还可以分为（max）和（min）等多项子属性。max 参数值可指定全文检索内存管理器在较大内存缓冲区中保持的最大缓冲区数。如果 max 的值为 0，则较大缓冲池中可以保持的缓冲区数就没有上限；min 参数可指定较大内存缓冲池中必须保持的最小内存缓冲区数。Microsoft SQL Server 内存管理器发出请求后，将释放所有额外的缓冲池，但将保留该最低数量的缓冲区。不过，如果指定的 min 值为 0，则释放所有内存缓冲区。

（2）fill factor：填充因子设置。最大值为 100，最小值为 0，系统默认为 0。

使用 fill factor 选项可以指定 Microsoft SQL Server 2005 使用现有数据创建新索引时将每页数据填满到多大程度。由于在页填充时 SQL Server 必须花时间来拆分页面，因此填充因子会影响性能。

在实际应用中，很少有需要手工更改 fill factor 的默认值，因为可以使用 CREATE INDEX 或 ALTER INDEX REBUILD 语句来覆盖其对于指定索引的值。值得注意的是，当 FILLFACTOR 设置为 0 或 100 时，将完全填充叶级别。

（3）max worker threads：设置最大工作者线程数量。其最小值为 0，最大值为 32767，系统默认值是 128。

使用 max worker threads 选项可以配置 Microsoft SQL Server 进程可使用的工作线程数。SQL Server 使用 Microsoft Windows 2000 和 Windows Server 2003 操作系统的本机线程服务，而线程池则处理所有用户。

该属性的用途在于当服务器上连接有大量客户端时，线程池有助于优化性能。一般情况下，为每一个客户端连接创建一个独立的操作系统线程可占用较少的系统资源。但是，当服务器上具有数以百计的连接时，每个连接使用一个线程就可能占用大量的系统资源。max worker threads 选项使 SQL Server 可以为大量的客户端连接创建一个工作线程池，这将提高性能。

如果 max worker threads 的默认值是 0，则允许 SQL Server 在启动时自动配置工作线程数。对于大多数系统而言，该设置为最佳设置；然而，根据自己的系统配置将 max worker threads 设置为特定值有时会提高性能。

【计算最大工作线程数】

微软建议对于 32 位 SQL Server，设置最大为 1024；对于 64 位 SQL Server，设置最大为 2048。更可以使用公式（256+（<处理器数>－4）×8）来计算 32 位 SQL Server 的线程数，而 64 位 SQL Server 的线程数设置为 32 位 SQL Server 数值的 2 倍即可。

（4）index create memory：该属性可控制最初为创建索引分配的最大内存量。其最小值为 0，系统默认值为 704，而最大值可以为 2147483647。需要注意的是，实际运行值不会超过用于运行 SQL Server 的操作系统和硬件平台的实际内存量：比如在 32 位操作系统中，运行值将小于 3GB。

如果设置该属性后，之后创建索引时需要更多内存，而且有内存可供使用，服务器将使用可用的内存，从而超出此选项的设置。如果没有内存可供使用，则继续使用已分配的内存来创建索引。

根据对 SQL Server 2005 中分区表和索引的介绍，如果出现非对齐的分区索引且并行度很高，则创建索引时的最低内存要求将显著提高。在 SQL Server 2005 中，此属性的设置可控制在单一索引创建操作中为所有索引分区分配的初始内存量。如果此选项设置的内存量小于运行查询所需的最小内存，查询将终止并显示错误消息。如果在创建索引时遇到困难，可以考虑提高此选项的运行值。

（5）locks：使用 locks 选项可以设置可用锁的最大数量，以便限制数据库引擎用于这些锁的内存量。默认设置为 0，即允许数据库引擎根据不断变化的系统要求动态地分配和收回锁结构，最小值为 5000，最大值为 2147483647。

如果服务器启动时 locks 设为 0，锁管理器将从数据库引擎中获取足够的内存，用于包含 2,500 个锁结构的初始池。当锁池用完时，将另外为该池获取内存。

通常情况下，如果锁池需要的内存比数据库引擎内存池中可用的内存少，而具有更多可用的计算机内存（尚未达到 max server memory 的阈值），则数据库引擎将动态分配内存以满足锁的需求。但是，如果内存分配导致操作系统级的换页（例如，如果另一个应用程序与 SQL Server 实例在同一台计算机上运行并使用该计算机的内存），则不会分配更多的锁空间。动态锁池获取的内存不会超过分配给数据库引擎的内存的 60%。如果锁池获取的内存达到了数据库引擎实例所获取内存的 60%，或计算机上没有更多的可用内存，则再发出针对锁的请求将生成错误。

【对锁配置的建议】

建议的配置为允许 SQL Server 动态地使用锁。但是，可以通过设置 locks 从而替代 SQL Server 动态分配锁资源的能力。将 locks 的值设为 0 以外的值后，数据库引擎分配的锁的数量不能大于在 locks 中指定的值。如果 SQL Server 显示消息说明超过了可用锁数，请增大该值。由于每一个锁都需要消耗内存（每一个锁需 96 字节），增加该值将增加服务器对内存的需要。

Locks 属性也会影响何时进行锁升级。当 locks 属性设为 0 时，则当前锁结构使用的内存达到数据库引擎内存池的 40%时将进行锁升级。如果 locks 未设为 0，则实际锁的数量达到 locks 的指定值的 40%时将进行锁升级。

（6）min Server memory 和 max Server Memory：这是两个对应的属性。使用 min server memory 和 max server memory 这两个服务器内存选项可以重新配置 Microsoft SQL Server 实例所使用的缓冲池的内存量（以 MB 为单位）。

默认情况下，SQL Server 的内存要求会根据可用系统资源的情况动态地变化。min server memory 的默认设置为 0，max server memory 的默认设置为 2147483647。可以为 max server memory 指定的最小内存量为 16 MB。

【SQL Server 使用内存规则】

当 SQL Server 动态使用内存时，它会定期查询系统以确定可用物理内存量。比如在 Microsoft Windows 2000 中，SQL Server 根据服务器的活动来增大或收缩缓冲区高速缓存，以使可用物理内存保持在 4 MB 到 10 MB 之间。保持此可用内存可避免 Windows 2000 分页。如果可用内存较少，则 SQL Server 将内存释放给 Windows 2000。如果可用内存较多，则 SQL Server 将内存分配给缓冲池。SQL Server 仅在其工作负荷需要较多内存时才向缓冲池增加内存；处于休眠状态的服务器不会增大其缓冲池的大小。

微软官方建议允许 SQL Server 动态使用内存；但需通过手动设置内存选项并限制 SQL Server 可以访问的内存量。在设置 SQL Server 使用的内存量之前，应确定适当的内存设置，具体方法是从总的物理内存中减去操作系统、其他程序（包括数据库的其他实例）所需内存，所得差值就是可以分配给 SQL Server 使用的最大内存量。

【设置使用最小值内存的问题】

如果将 max server memory 设置为最小值，那么可能会严重降低 SQL Server 的性能，甚至使其无法启动，不过这种情况在实际工作中比较少见。如果在更改此选项之后无法启动 SQL Server，则可以使用-f startup 开关进行启动，再将 max server memory 恢复为之前的正常值。

（7）min memory per query：每个查询使用最小内存数属性。使用 min memory per query 选项指定分配给查询执行时所需要的最小内存量（KB）。例如，如果将 min memory per query 设置为 2048 KB，则查询保证将至少获取那么多的总内存。可以将 min memory per query 设置为从 512 到 2 147 483 647 KB（2 GB）的任何值。该属性的系统默认值为 1024 KB。

在 SQL Server 中，数据库系统内的查询处理器来确定要分配给查询的最佳内存量。min memory per query 属性允许管理员通过手工设置指定任何单个查询收到的最小内存量。如果查询需要对大量数据执行哈希（Hash）和排序（Sort）操作，则这些查询获得的内存通常比该选项指定的最小内存多。因此，对于一些小型查询和中等大小的查询，增大 min memory per query 的值可能会提高性能，但会导致内存资源争夺加剧。

（8）user connections：该属性设置用户连接的限值：使用 user connections 选项可以指定 Microsoft SQL Server 上允许同时建立的最大用户连接数。实际允许的用户连接数还取决于正使用的 SQL Server 版本以及应用程序和硬件的限制。SQL Server 允许的最大用户连接数为 32767。

由于 user connections 是动态（自动配置）选项，SQL Server 将根据需要自动调整最大用户连接数，最大不超过允许的最大值。例如，如果仅有 10 个用户登录，则要分配 10 个用户连接对象。在大多数情况下，没有必要改变该选项的值。使用 sp_configure 可以确定系统允许的最大用户连接数。

【限值用户连接的好处】

使用 user connections 选项有助于避免由于过多并发连接而使服务器超载。可以根据系统和用户要求估计连接数。例如，在很多用户的系统上，每个用户通常不要求唯一的连接。可以在用户间共享连接。对于运行 OLE DB 应用程序的用户，每个打开的连接对象需要一个连接；对于运行开放式数据库连接（ODBC）应用程序的用户，每个活动连接句柄需要一个连接；对于运行 DB-Library 应用程序的用户，每个调用 DB-Library dbopen 函数的启用的进程需要一个连接。

【设置用户连接的注意事项】

若必须手工设置用户连接这个属性，则不要将值设置得太高。这是因为：不论是否使用当前连接，每个连接都至少需要大约 28 KB 的开销。而如果超过了用户连接的最大允许值，将收到一条错误消息，连接将会断开，而且这样的情况直到出现下一个可用连接之后才能重新建立。

除前文所述诸多属性之外，SQL Server 2005 中还包含若干安全、性能、工作方式等相关的设置，由于更少用到，就不再列出了，感兴趣的读者可以查看 MSDN 进一步了解。

总之，数据库系统是 Web 应用绝大部分数据存放的位置，作为 Web 应用的最后端，如何快速地存取信息确实是需要所有相关人员、特别是开发人员和性能测试人员高度重视的问题，需要在实际工作中不断地积累经验。

17.4　结　束　语

本书讲解到这里就全部结束了。在本书中，我们知道软件、硬件都可能对 Web 应用的性能产生影响，要结合起来分析。对于硬件，开始的服务器选型很重要，它能够让一个 Web 应用赢在起跑线上。而对于软件影响，可以使用功能强大的 LoadRunner 或者其他的性能测试软件，结合系统和应用服务器提供的性能计数器，各厂商提供的免费或者收费工具进行性能的量度与测试，最终发现性能问题所在。只有发现问题的原因之后，进一步的性能优化才会有的放矢。

由于本人才疏学浅，另外也限于篇幅等原因，本书只介绍了性能测试的基础和初中级的部分。俗话说：千里之行，始于足下。读者应在性能测试的实际应用中继续地学习、探索，在遇到困难的时候多查阅官方文档，多去优秀的测试社区中请教。并且，作为性能测试工程师，特别要注意多学习操作系统、数据结构以及程序设计等学科的知识，最终一定会成为性能测试的高手。

一切皆有可能。

附录 A　主要性能测试工具下载网址

本附录列出了主要性能测试工具（除了本书介绍的 LoadRunner）的下载网址，并在软件说明中列出了可下载版本的种类（免费/共享软件或试用版本软件），读者可以在决定使用何种具体工具的时候做为参考。

软 件 名 称	网　　址	简介与可下载版本
Advanced .NET Test System (Ants) Performance Profiler	www.red-gate.com	对在.NET平台下开发的软件进行测试的工具，包含内存泄漏检测、行级代码性能分析等功能。网站可下载14天全功能的试用版
Benchmark Factory	http://www.quest.com/benchmark-factory/	通过对不同情况下的环境进行Benchmark打分，来确定系统的可扩展性，为发现当前系统的极限，部署高效应用之前的选型服务。对于数据库的性能测试很有用处。可下载试用版
BurnIn Test	www.passmark.com	对机器内各硬件进行性能测试和评分的小型工具软件，可下载试用版
Forecast	http://www.facilita.co.uk/	支持Web和.NET、Java等平台的性能测试工具，与LoadRunner测试过程类似。另有网络仿真套件，支持多种网络协议和环境。可下载试用版
JMeter	http://jakarta.apache.org/jmeter/	Apache下的JMeter 可以用于对静态、动态资源（包括文件、Servlet、Perl脚本、Java 对象、数据库查询、FTP服务器等）的性能进行测试。JMeter还包含性能的图形分析功能，支持多线程、大并发的性能测试。免费
OpenSTA	http://opensta.org/	开源的对Web应用进行性能测试工具
QALoad	http://www.microfocus.com/products/QALoad/index.asp	原由Compuware公司开发，目前转为MicroFocus公司产品的企业性能测试工具软件，支持多种平台
Web Performance Suite	http://www.webperformanceinc.com/load_testing/	专业Web性能测试公司的一系列性能测试产品，支持自动化性能测试，脚本编写简单

续表

软 件 名 称	网　　址	简介与可下载版本
SilkPerformer	http://www.borland.com/us/rc/lifecycle-quality-management/software-performance-testing.html	Borland公司的性能测试工具，专长于Citrix平台。可下载30天试用版
Rational Suite Performance Studio	http://www.ibm.com/software/rational/	IBM公司出品的专业性能测试工具，是其强大的Rational套件的一部分
Web Server stress Tool	http://www.paessler.com/	专业网络分析公司的Web Server性能测试工具。支持模拟每小时100万以上的页面浏览和1万以上的并发用户
Webload	http://www.radview.com/	专业的Web性能测试工具，支持Flex的测试，可下载试用版

附录 B　部分性能测试网站列表

性能测试属于软件测试的一个分支，为了学习方便，本附录列出了部分测试相关网站的网址，中文和英文都有，供读者参考。

网 站 名 称	网　　址	内 容 简 介
软件测试网	www.51testing.com	国内最好的测试网站之一，有专门的软件测试培训、论坛和博客等
测试时代	www.ltesting.com	国内最好的测试网站之一，有专门的性能测试频道，首页包括了很多种测试的链接，一目了然。另有测试培训、测试电子杂志等
测试论坛 QAForum	www.qaforum.com	英文网站。一个非常专业的测试论坛，包含各类测试以及测试工具的讨论，可以说是把握最新测试技术的最好窗口之一
微软测试中心	http://msdn.microsoft.com/en-us/testing/default.aspx	微软官方网站中的测试技术中心，对于了解微软公司对于软件测试理念和方法的理解很有益处
SQATester	www.sqatester.com	一个包含各种技术文档、面向质量保障（QA）的测试网站，内容较容易阅读
测试流程介绍 MSDN	http://msdn.microsoft.com/en-us/library/bb671346.aspx	英文网页。包含多个微软内部专业人士讲解性能测试主要流程的英文视频
软件测试Wiki	http://en.wikipedia.org/wiki/Software_testing	对于了解软件测试方面的概念非常好的网站，可以说是一个软件测试的图书馆